CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Nil Martins de Moura

INTERFACE WEB CONTROLADORA DE UM BRAÇO ROBÓTICO

Timóteo

2017

Nil Martins de Moura

INTERFACE WEB CONTROLADORA DE UM BRAÇO ROBÓTICO

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Elder de Oliveira Rodrigues

Timóteo

2017

NIL MARTINS DE MOURA

INTERFACE WEB CONTROLADORA DE UM BRAÇO ROBÓTICO

Monografia apresentada à coordenação de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais Campus Timóteo, como parte das exigências para a obtenção do grau de Bacharel em Engenharia de Computação.

Local, 18 de Agosto de 2017.

BANCA EXAMINADORA

Prof. Dr. Elder de Oliveira Rodrigues
(Orientador)

Prof. Dr. Viviane Cota Silva
(Professor Convidado)

Prof. Dr. Rodrigo Gaiba de Oliveira
(Professor Convidado)

Dedico este trabalho à minha família,
em especial à meus pais, Cheslei e Márcia,
minhas irmãs Letícia e Mariana, meu irmão Kleyver
e minha namorada Sâmella, que em todo tempo me apoiaram
e me incentivaram para a busca desta nova conquista.

Agradecimentos

Agradeço primeiramente a Deus pois sou inteiramente dependente Dele. Sem seu cuidado, seu amor e sua graça eu não teria chegado até aqui. Durante esse tempo de graduação pude aprender tantas coisas, foram tantos momentos, que fez com que o meu coração fosse cada vez mais grato à esse Deus, o único e verdadeiro, o dono de todo conhecimento, a sabedoria em essência.

Agradeço com muita alegria a minha família. Foi uma caminhada difícil, durante esse tempo eu fui o filho mais distante, na maioria das vezes estudando. Em todos os momentos pude contar com o apoio e seus conselhos. Em especial quero agradecer a meus pais, Cheslei e Márcia e a minha vó, Iza, muito obrigado pelo suporte e por tudo que fizeram por mim, vocês são co-vitoriosos comigo.

Quero agradecer a minha namorada, Sâmella, que sempre me ajudou e apoiou. Obrigado pela compreensão, pelo cuidado, carinho e por sempre estar ao meu lado. Agradeço também a sua família, a qual sou de mesmo modo grato.

Presto meus agradecimentos a meus colegas de turma, parceiros em momentos de dificuldade e árduo labor (sorrisos). Também a meus professores, que proveram orientações necessárias para concluir e obter esta vitória. Sou grato por todo conhecimento que me passaram.

Agradeço em especial a meu orientador, Elder de Oliveira Rodrigues pela paciência e companheirismo. Obrigado por sempre ser prestativo, e por me instruir da melhor maneira possível.

Um grande abraço e obrigado!

“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Nesse trabalho propomos uma implementação de uma interface web controladora, que consiste em um sistema web e um protocolo de comunicação Universal Serial Bus (USB) 2.0. Tais componentes foram desenvolvidos com o intuito de ser produtos de um resultado inicial, com a possibilidade de ser posteriormente continuado e/ou aprimorado, permitindo então, o controle efetivo de um braço robótico. Durante os capítulos seguintes deste trabalho será apresentado e discutido todo um conjunto básico de conhecimentos fundamentais para o uso e aprimoramento do mesmo. Antes de se iniciar a produção dessa interface, faz-se necessário a definição das linguagens de programação, frameworks, APIs, ferramentas de desenvolvimento e quais técnicas de Engenharia de Software serão necessárias para a produção do sistema web. Para o desenvolvimento do protocolo de comunicação, foi imprescindível a utilização de um microcontrolador, linguagem de programação e ferramentas de desenvolvimento e simulação. Como resultado obteve-se um sistema web com capacidade de executar comandos de movimentação dos eixos dos servomotores via protocolo de comunicação.

Palavras-chave: sistema web, braço robótico, controladora, protocolo de comunicação.

Abstract

In this work we propose an implementation of a web controller interface, which consists of a web system and a Universal Serial Bus (USB) 2.0 communication protocol. These components were developed with the intention of being products of an initial result, with the possibility of being later continued and/or improved, thus allowing the effective control of a robotic arm. During the following chapters of this work will be presented and discussed a whole basic set of fundamental knowledge for the use and improvement of it. Before starting the production of this interface, it is necessary to define the programming languages, frameworks, APIs, development tools and which Software Engineering techniques will be necessary for the production of the web system. For the development of the communication protocol, it was essential to use a microcontroller, programming language and development and simulation tools. As a result, we obtained a web system capable of executing commands to move the servomotors axes via communication protocol.

Keywords: web system, robotic arm, controller, communication protocol.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Interface de Programação de Aplicações
CCP	Capture/Compare/PWM
DER	Diagrama de Entidade e Relacionamentos
USB	Universal Serial Bus
HID	Human Interface Device Class
IA	Inteligência Artificial
IDE	Ambiente de Desenvolvimento Integrado
IHC	Model View Controller
UML	Linguagem de Modelagem Unificada
MVC	Model View Controller
ORM	Mapeamento Objeto-Relacional
PIC	Controlador de Interface Programável
PWM	Pulse Width Modulation
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

Sumário

1 Introdução

“Não existe fracasso. Existem somente resultados.”.
Anthony Robbins

Podemos observar na sociedade atual uma grande necessidade de produzir, construir e realizar soluções cada vez mais eficientes e precisas para um determinado problema. Para se chegar nesses tipos de soluções, se faz cada vez mais necessário o uso de dispositivos automáticos, programados para realizar determinadas tarefas, que muitas vezes, nós humanos não conseguimos desenvolver, pelo alto grau de dificuldade, repetibilidade e periculosidade, ou até mesmo, por falta de eficiência no realizar o demandado. Tais dispositivos são estudados e desenvolvidos através da Robótica. De forma usual, o termo Robótica é formalmente empregado para definir a disciplina associada ao uso e programação de robôs, e a expressão Engenharia Robótica, ainda mais específico, refere-se à construção de robôs e dispositivos robóticos (SANTOS, 2004).

Com o avanço tecnológico e por meio desse, a evolução e miniaturização dos circuitos integrados, a robótica vem crescendo, expandindo horizontes e se tornando cada vez mais comum e extremamente necessária em nosso dia a dia, vale ressaltar que a mesma é uma área de pesquisa multidisciplinar (POZZEBON, 2013 apud JONES; FLYNN, 1993), dando a possibilidade de desenvolver competências nas diversas áreas de aprendizagem. Nesse cenário amplo da robótica, surge então o conceito, Robótica Educacional ou Robótica Pedagógica, que segundo SILVA (2009), é o ambiente de aprendizagem de interação entre o aluno e professor, em que o mesmo direciona conhecimentos específicos da robótica, que vão desde a montagem dos robôs até o controle dos mesmos através de computadores.

Adicionalmente, segundo SILVA (2009) "o casamento entre a robótica e educação tem todos os ingredientes para dar certo. Primeiro, o robô, como elemento tecnológico, possui uma série de conceitos científicos cujos princípios básicos são abordados pela escola. Segundo, pelo fato de que os robôs mexem com o imaginário infantil, criando novas formas de interação, e exigindo uma nova maneira de lidar com símbolos". E estamos realmente vivenciado isso. Nessa nova possibilidade de aprender e adquirir novos conhecimentos, o aluno tem um papel diferente do comum, a construção de conhecimentos sucede através de suas próprias observações e experiências, tendo um significado maior (MAISONNETTE, 2002) e uma menor dissonância cognitiva (FESTINGER, 1975).

Segundo Zilli (2004), a robótica possibilita ao estudante tomar conhecimento da tecnologia atual, desenvolver habilidades e competências, como: trabalho de pesquisa, a capacidade crítica, o senso de saber contornar as dificuldades na resolução de problemas e o desenvolvimento do raciocínio lógico. É certo que a robótica vem se mostrando como um potencial de elevar o nível de qualidade da educação nas escolas.

1.1 Justificativa

Este trabalho justifica-se pela necessidade de se implementar um sistema web para controle e gerenciamento de um braço robótico de um kit existente, com o intuito de que o mesmo futuramente possa ser utilizado como um material mediador de ensino-aprendizagem nas escolas. Como motivação teve-se a vontade de compreender e desenvolver um protocolo de comunicação USB em um chip microcontrolador, bem como o interesse de produzir um sistema que com ele se comunique.

1.2 Problema

A seguinte questão constitui o problema deste trabalho: A interface web proposta foi preparada com o objetivo de poder ser futuramente continuada em um outro trabalho de conclusão de curso e assim controlar o kit Robix Rascal. Seguindo o que foi descrito anteriormente, questiona-se se essa implementação proposta poderá ser eficaz em cumprir os seus objetivos.

1.3 Objetivos

Para responder ao problema proposto, o objetivo geral deste trabalho é propor um sistema web de interação com o usuário, preparando-o para uma controladora de um braço robótico.

Também, objetivam-se mais especificamente:

1. Identificar as limitações da implementação apresentada nesse trabalho;
2. Propor uma implementação de protocolo de comunicação: execução de tarefas passo a passo e execução instantânea.

2 Procedimentos metodológicos

“Um bom começo é a metade”.
Aristóteles

O presente trabalho é de caráter descritivo e exploratório do ponto de vista dos objetivos; aplicado do ponto de vista de sua natureza; qualitativo quanto à abordagem ao problema; pode ser classificado como pesquisa experimental na perspectiva de se produzir um sistema preparado para acionar uma controladora para o movimento de um braço robótico, identificando as possíveis limitações desse.

Os procedimentos metodológicos são organizados nas seguintes etapas:

1. pesquisar e estudar trabalhos de robótica (móvel e educativa) referentes às tarefas que deverão ser realizadas nesse trabalho em específico;
2. definir e estudar as possíveis tecnologias que serão utilizadas na implementação do sistema web e do protocolo de comunicação;
3. projetar e implementar o sistema web de interação do usuário;
4. projetar e implementar um protocolo de comunicação que permitirá a interação entre o sistema e a controladora;

2.1 Revisão da literatura

A revisão de literatura empregada neste trabalho resulta em dois conjuntos distintos de trabalhos relacionados:

- a revisão do estado da arte em robótica móvel e educativa;
- o levantamento bibliográfico dos principais resultados de trabalhos que explicitam conceitos e aplicações referentes ao estado da arte desse.

O estado da arte é apresentado no capítulo 3. O contexto educativo foi adotado para limitar o escopo deste trabalho e torná-lo viável para futuro estudo com usuários (alunos), sendo possível o uso do mesmo em experiências práticas laboratoriais. Com isso, trabalhos relacionados mais especificamente à robótica educativa são especialmente de interesse. Na literatura sobre robótica também se buscam trabalhos que retratam robótica móvel, mais especificadamente, a existente nos braços robóticos. Características da robótica educativa naturalmente permeiam os produtos tecnológicos de robótica nas escolas e são discutidas em trabalhos sobre o tema nas áreas de Ciência da Computação, Engenharia Elétrica, Engenharia da Computação, Ciência da Informação, e em algumas áreas de estudo da cognição do aprender.

2.2 Definição e estudo das possíveis tecnologias à utilizar

Deseja-se ter como produto final um sistema web de interação preparado para uma controladora, que poderá ser produzida posteriormente em um outro trabalho. Antes de se iniciar a produção desse sistema, deve-se definir as linguagens de programação, frameworks, APIs, ferramentas de desenvolvimento e quais técnicas de Engenharia de software serão necessárias. Para o desenvolvimento do protocolo de comunicação, definir o microntrolador, a linguagem de programação e quais ferramentas de desenvolvimento e simulação.

2.3 Projeto e implementação de um sistema de interação

Este projeto descreve os requisitos, e a formulação de diagramas DER e UML, ambos técnicas de Engenharia de Software comumente usadas.

Para o desenvolvimento foi escolhido Python web e Javascript como linguagens de programação, Django como framework de desenvolvimento web; Jquery, AngularJS e Bootstrap como APIs; Visual Studio 2015 Community como ferramenta de desenvolvimento.

Desta forma, o resultado desse projeto proporcionará a interação entre o usuário e o sistema web, a permitir o gerenciamento de tarefas do braço robótico, como movimentar e executar rotinas etc.

2.4 Projeto e implementação da comunicação entre o sistema de interação e a controladora

Para a interação entre o sistema web e a controladora será necessário o desenvolvimento de um protocolo de comunicação USB, que será produzido para ambos os lados da comunicação. Para o lado do sistema, será utilizado uma biblioteca Python de comunicação USB. Para o lado da controladora, a linguagem C, um gravador de programas específico para microcontroladores, ferramentas de desenvolvimento e componentes eletrônicos para a montagem do circuito que será proposto.

Como resultado, ambos os lados estarão operando com o protocolo, permitindo a comunicação USB entre eles.

3 Fundamentos teóricos e históricos

'Procure ser um homem de valor, em vez de ser um homem de sucesso'
Albert Einstein

O principal objetivo deste capítulo é apresentar os fundamentos históricos e teóricos sobre os quais este trabalho é executado, além de expor de maneira breve os principais e mais recentes trabalhos das áreas de robótica móvel e educativa.

3.1 Fundamentos teóricos

Para fundamentar o caminho que será percorrido até alcançar os objetivos finais, nos próximos sub-tópicos serão detalhados e descritos conceitos teóricos, com o intuído de embasar e nortear o desenvolvimento do mesmo.

3.1.1 Robótica

A Robótica pode ser definida como "a ciência dos sistemas que interagem com o mundo real com pouca ou mesma nenhuma intervenção humana" (CONSULT, 1995). É um ramo da tecnologia bastante abrangente, que engloba áreas da mecânica, eletrônica e computação, criando um nexo, de maneira harmoniosa, entre as mesmas. Atualmente, a robótica trata de sistemas compostos por máquinas ou partes mecânicas automáticas controladas por circuitos integrados programáveis ou não programáveis, tornando sistemas mecânicos motorizados, e controlados manualmente ou automaticamente por circuitos elétricos. Tais máquinas executam movimentos, tomam decisões, analisam, mas são apenas uma imitação da vida humana, são apenas robôs (OTTONI, 2010).

Segundo Silva (2009), a criação e o desenvolvimento mecânico, elétrico e computacional do robô só ocorreu no século XX, mas o desejo e o sonho de criar esse ser artificial e o vivificar acompanha o homem há séculos (PAZOS, 2002).

O termo robô surge mais tarde, no século XX, derivando da palavra tcheca robota que significa trabalhador forçado (ou escravo). O termo, com a sua atual interpretação, foi inventado pelo escritor tcheco Karel Capek em seu romance R.U.R. ("Robôs Universais de Rossum"), em 1921. Basicamente, o romance retrata um personagem chamado Rossum que projeta e constrói um exército de robôs que se tornam cada vez mais inteligentes e com objetivo de dominar o mundo. O foco principal da obra é a extinção do homem face a um meio tecnológico, o que inspirou inúmeros enredos de ficção científica. O termo robótica também saiu da literatura quando em 1941, o escritor russo-americano Isaac Asimov (1920-1992) escreveu um conto intitulado "Runaround", em que o termo significa o estudo e o uso de robôs. Mais tarde o termo foi adotado pela comunidade científica. Entretanto, a robótica nasceu por meio da ficção científica, mas não é a mesma. É uma ciência em expansão e transdisciplinar por

natureza, envolvendo várias áreas de conhecimento, tais como: microeletrônica, computação, engenharia mecânica, inteligência artificial (IA), física (cinemática), neurociência, entre outras (HALFPAP et al., 2005). Portanto, a robótica é a ciência ou o estudo da tecnologia associado com o projeto, fabricação, teoria e aplicação dos robôs (SILVA, 2009).

Segundo Russell, Norvig e Souza (2004), robôs são agentes físicos, que executam tarefas manipulando o mundo material, ver figura 1. Para essa execução, esses agentes são equipados com atuadores (pernas, rodas, articulações e garras), que exercem força física sobre o mundo, e com sensores, que permitem perceber o ambiente em que está inserido. Idealmente, um robô deve ter os seguintes elementos:

1. Atuadores: são meios pelos quais os robôs se movem e alteram a forma de seus corpos. Por exemplo, braços, pernas, mãos, pés;
2. Sensores: peças que funcionam como sentidos que podem detectar objetos, calor ou luz; depois converte a informação em símbolos processados por computadores; Por exemplo, sensores de cor, de fim de curso, de som;
3. Computador: o "cérebro" que contém instruções, isto é algoritmos, para controlar o robô; por exemplo, microcontroladores;
4. Equipamentos ou mecanismos: isso inclui ferramentas e equipamentos mecânicos. por exemplo, rodas, correias.

Figura 1 – Robô soldado TALON (utilizado pelo exército dos EUA no Iraque) [1]

Fonte: http://vitaminanerd.com.br

Nem todas as máquinas podem ser consideradas robôs, muitas vezes o que difere um robô de uma máquina comum, é que o primeiro quase sempre funcionam por si só, são

[1] Figura 1:
Disponível em: http://vitaminanerd.com.br/que-robos-sao-utilizados-por-forcas-militares
Acesso em Out. 2016

sensíveis ao seu ambiente, adaptam-se às suas variações, ou a erros no desempenho anterior, são orientados para tarefas, existindo ainda a possibilidade do mesmo realizar uma tarefa utilizando diferentes métodos; o que depende exclusivamente de suas habilidades.

Atualmente, existem três categorias pelas quais os robôs podem se categorizar, são elas: manipuladores, robôs móveis e híbridos. Os manipuladores são fixos ao local onde realiza o trabalho, enquanto os móveis se deslocam em seu ambiente com a utilização dos sensores e atuadores. Os híbridos são basicamente a junção dos dois anteriores (RUSSELL; NORVIG; SOUZA, 2004).

Para o Instituto de Robótica da América (R.I.A.), robô é um manipulador, re-programável e multi-funcional, projetado para mover materiais, partes, ferramentas ou dispositivos especializados através de movimentos variáveis, podendo ser programados para desempenhar uma variedade de tarefas.

Segundo Silva (2009) a robótica é uma ciência nova e seu campo de atuação se multiplica com muita rapidez. Máquinas robotizadas têm sido usadas nas indústrias automobilísticas, realizando tarefas como pintura ou montagem; assim como, em outras indústrias, como de eletrodomésticos, eletro-eletrônica, música, alimentícia, têxtil, calçados, petrolífera, entre outras; o uso da robótica é de extrema importância e traze muitos benefícios, existindo em muitos casos uma certa dependência de tais tecnologias.

Adicionalmente, Pazos (2002) explicitou algumas razões para a utilização de robôs na indústria, a saber:

1. Custo: o custo de um robô é mais barato do que de um operário, sem considerar que um robô pode trabalhar em 95% do tempo da tarefa a ser realizada;
2. Produtividade: em algumas aplicações, os robôs podem trabalhar mais rápido que os humanos, sem falar na economia de material;
3. Qualidade do produto: devido às capacidades de maior precisão e velocidade, um robô faz um produto com melhor qualidade e em menos tempo;
4. Operatividade e adaptatividade: devido ser uma máquina programável, o mesmo tem capacidade de operar em ambientes hostis ou com materiais perigosos.

Porém, não são apenas no ramo industrial que os robôs são utilizados. Comumente vemos que o uso dos mesmos tem extrapolado para diversas áreas. Serão apresentados abaixo exemplos de aplicação em algumas áreas (SILVA, 2009).

1. Robôs domésticos: Destinados ao uso doméstico, esses robôs são programados para executar trabalhos domésticos, atuar como vigilantes, ou até mesmo ser um mascote do usuário. Um dos robôs domésticos mais famosos é o robô cão *AIBO* da *Sony*, mostrado na figura 2a.
2. Robôs de entretenimento e robôs Sociais: São projetados para o lazer ou para entreter pessoas. Na sua grande maioria, são humanoides ou antropomórficos, como um ca-

chorro ou gato. Podem ser guias de museus e exposições, acompanhantes ou mesmo, brinquedos. Um exemplo é o robô *Enon*, mostrado na figura 2b, da *Fujitsu*, que é utilizado como guia no *Museu Kyotaro Nishimura*, em *Tóquio*.

3. Robôs médicos: Os robôs-cirurgiões aumentam o desempenho do cirurgião, conseguindo uma precisão de movimentos muito superior, efetuando cortes mínimos. Além de ser usado em ambiente hospitalar, o mesmo pode servir para treinamento e ensino em ambientes virtuais no estudo de medicina. Um exemplo é o robô *Da Vinci*, figura 2d, um robô portátil, que possui quatro braços robóticos interativos e pode operar e diagnosticar pacientes.

4. Robôs militares: Os robôs militares são utilizados em confrontos com civis ou militares. Existem inúmeras aplicações no ramo, espionagem em território inimigo é uma delas. Um exemplo é o *T-Hawk*, mostrado na figura 2c, um robô do tipo veículo aéreo não tripulado (UAV), usado pelo *Exército dos Estados Unidos da América* para reconhecimento e vigilância durante operações.

5. Veículos autônomos inteligentes e AGVs: *AGV (Automated Guided Vehicle)* é um robô que tem várias utilidades, entre elas, a mais comum é o transporte de cargas pesadas ou materiais de alta periculosidade (MURPHY, 2000). Os veículos autônomos são robôs aplicados à condução e ou vigilância. Esses robôs interagem com o ambiente ao seu redor mapeando o mesmo, obtendo informações, tomando decisões através dessas. Um exemplo é o *Kiva*, mostrado na figura 2e, um robô AGV para movimentação de cargas em armazéns. Os mesmos são produzidos pela *Amazon*. Um exemplo de aplicação, em empresas de comércio eletrônico, pode ser visto na própria Amazon; Na mesma, tais robôs movimentam com as prateleiras que situam os produtos adquiridos em uma determinada compra até o embalador, para que ele retire o produto e redirecione o mesmo para as próximas etapas de logística.

6. Robôs de busca e salvamento: São robôs projetados para realizar buscas e salvamentos, em terra, no ar ou na água. Utilizados para resgates de civis em incêndios, terremotos e outras catástrofes. O robô humanoide *BEAR*, mostrado na figura 2f, foi projetado para localizar e transportar civis em situação de risco.

(a) Aibo: Robô Doméstico.

(b) Enon: Robô Guia.

(c) T-Hawk: Robô Militar.

(d) Da Vinci: Robô Médico.

(e) Kiva: Robô Autônomo.

(f) Bear: Robô Humanoide

Figura 2 – Robôs em diferentes áreas [2].

Esses foram alguns exemplos que mostram o quanto a robótica é utilizada, a sua importância, e a sua capacidade de ser expansível a diversos horizontes.

3.1.2 Robótica Educacional

Também chamada de Robótica Pedagógica, a Robótica Educacional é existente em ambientes de aprendizagem em que o professor ensina ao aluno a montagem, automação e controle de dispositivos mecânicos que podem ser controlados pelo computador. A importância da robótica Educacional na educação está pelo fato da mesma abranger um processo de

[2] Figura 2a:
Disponível em: https://www.dramafever.com/pt/news/dead-robot-dogs-can-now-get-a-proper-buddhist-funeral/.
Acesso em Set. 2016;
Figura 2b:
Disponível em: http://www.thefutureofthings.com/5191-fujitsus-enon-robot/.
Acesso em Set. 2016;
Figura 2c:
Disponível em: http://www.vitaminanerd.com.br/que-robos-sao-utilizados-por-forcas-militares/.
Acesso em Set. 2016;
Figura 2d:
Disponível em: http://www.drugdangers.com/da-vinci/robot-lawsuit.htm/.
Acesso em Set. 2016;
Figura 2e:
Disponível em: http://www.fastcompany.com/1754454/kiva-powers-web-commerce-new-bot-bot-action/.
Acesso em Set. 2016;
Figura 2f:
Disponível em: http://www.roboticstoday.com/robots/bear-pictures.
Acesso em Set. 2016.

aprendizagem com atributos motivacionais, de colaboração entre as partes envolvidas (professores e alunos), de construção e reconstrução. Para isso, faz-se necessário a utilização de conceitos de diversas disciplinas para a construção de modelos, levando os alunos a uma rica vivência interdisciplinar. Assim, na construção de um modelo robótico, o processo de colaboração acontece quando os problemas são identificados e resolvidos em equipe e a autonomia é exercida na medida em que cada componente do grupo tem responsabilidade por uma parte da solução final, pelo seu próprio conhecimento e pelo grupo. Todos devem participar e contribuir para os resultados. Assim, as dificuldades de um integrante serão supridas por outro, fazendo com que o grupo cresça e se desenvolva mutuamente (SILVA, 2009).

A utilização de robôs como mediador para construção do conhecimento não é algo muito novo. O grande precursor desta atividade foi *Saymourt Papert*, pesquisador do MIT (Instituto de Tecnologia de Massachusetts). Seus trabalhos acerca da robótica na educação começaram nos anos 60 quando também nascia o construcionismo [Papert 1994]. Papert via no computador possibilidades presentes em seus recursos que poderia atrair as crianças e com isso facilitaria o processo de aprendizagem. Um de seus trabalhos mais célebres é a criação da linguagem LOGO. Essa linguagem tinha como elemento principal uma tartaruga, que inicialmente era um robô móvel que se deslocava no chão. Com o desenvolvimento do monitor de vídeo passou a ser representado de forma icônica na interface do programa onde se escrevia os comandos. E, da junção do LOGO com os brinquedos da LEGO, surgiu o sistema de robótica educacional (SRA) LEGO-LOGO. Com esse sistema, as crianças têm a possibilidade de construir seus protótipos e construir programas em LOGO para proporcionar comportamentos aos protótipos montados (SILVA, 2009). Através desse novo sistema, surgiram vários kits robóticos educativos, o mais popular deles é o *Lego Mindstorms*, que é fruto da parceria entre o *MIT Media Laboratory* e a Lego. Papert foi um dos apoiadores. A versão mais atual e a mais antiga pode ser vista nas figuras 3a e 3b respectivamente.

(a) Kit Lego Mindstorms EV3 (b) Kit Lego Mindstorms RCX

Figura 3 – Kits Lego Mindstorms [3].

[3] Figura 3a:
Disponível em: http://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313.
Figura 3b:
Disponível em: https://images-na.ssl-images-amazon.com/images/I/51T7B88ZNNL._SX425_.jpg.
Acesso em Set. 2016

3.1.3 Motores Elétricos

Comumente, os robôs de médio e pequeno porte utilizam acionadores elétricos para a realização de seus movimentos, tais acionadores são geralmente motores de corrente contínua (DC), motores de passo ou servo-motores. Estes motores não conseguem proporcionar velocidade e ou potência comparado aos acionadores hidráulicos, porém possibilitam maior precisão sobre as demais classes de acionadores (VOLK et al., 2005).

- Motores de passo: segundo Santos (2008) são dispositivos eletromecânicos que convertem pulsos elétricos em movimentos angulares discretos. O eixo de um motor de passo é rotacionado em pequenas unidades angulares, denominadas "passos", quando pulsos elétricos são aplicados em seus terminais. Tais pulsos possuem uma sequência, uma frequência e uma intensidade (número de pulsos em uma determinada fração de tempo); a sequência reflete na direção a qual o motor gira, a frequência reflete na velocidade do giro, e a intensidade representa o tamanho do ângulo rotacionado. Um motor de passo é sempre uma boa escolha quando movimentos demandados necessitam de precisão;
- Servo Motor: uma máquina eletromecânica, sua construção é dada sobre um motor DC incluindo um redutor de velocidade em conjunto com um sensor de posição e um sistema de controle realimentado, ver figura 4. Existem exceções, mas normalmente o eixo de um servo motor possui liberdade de apenas cerca de 180 graus (VOLK et al., 2005).

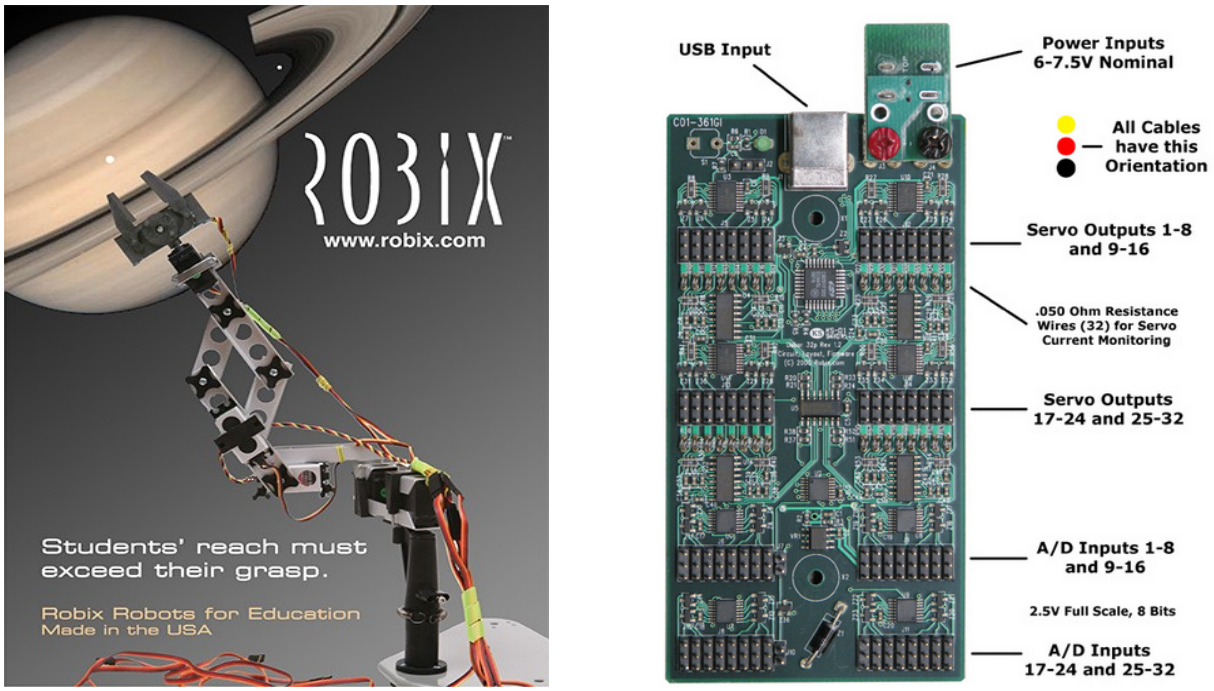
Figura 4 – Servo Motor Hitec HS-422[4]

Fonte: http://hitecrcd.com

[4] Figura 4:
Disponível em: http://hitecrcd.com/products/servos
Acesso em Set. 2016

3.1.4 Robix^{TM} Rascal RCS-6

O Robix^{TM} Rascal RCS-6, apresentado na figura 5, é um kit robótico educativo que contém basicamente todos os componentes para a montagem de um protótipo de braço mecânico. O mesmo possui além dos componentes para a estruturação do braço, seis servos motores e uma interface (eletrônica) controladora com sua devida fonte. Tal interface recebe e envia dados ao computador, codifica os movimentos em pulsos elétricos para movimentar os servo-motores, que por sua vez se movimentam nos dois sentidos. O movimento pode ser no sentido negativo (-) ou no positivo (+). A posição varia de -1400 a +1400.



(a) Braço Robótico (b) Interface Eletrônica do Robix

Figura 5 – Robix^{TM} Rascal RCS-6 [5].

3.1.5 Microcontrolador PIC

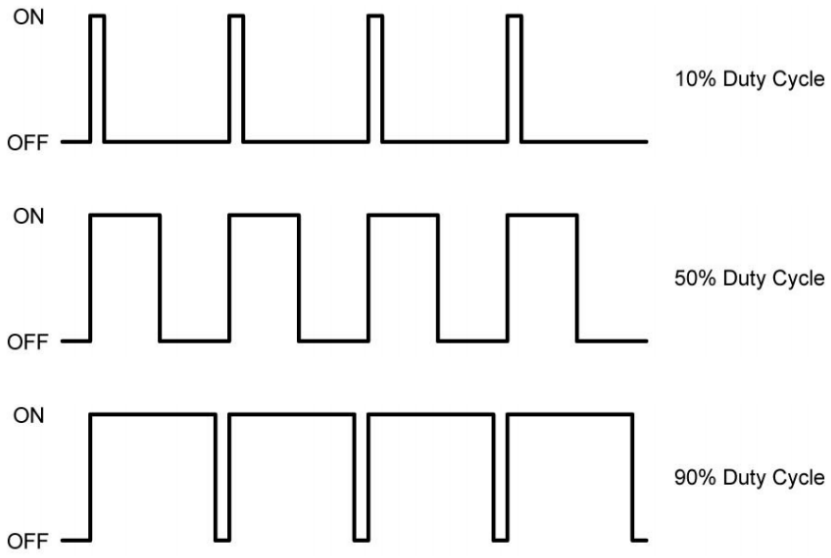
São microcontroladores que processam dados de 8, 16 ou 32 bits, com conjunto de instruções RISC. O mesmo é fabricado pela *Microchip Technology*, construído sobre a arquitetura de *Harvard*. O nome PIC é oriundo de "Controlador de Interface Programável" e representa uma das linhas de microcontroladores da fabricante. Tal linha é subcategorizada em diversos modelos de acordo com o tamanho do dado que o mesmo processa e seus periféricos internos. Em especial o PIC18F4550 processa dados de 8 bits, possui 40 pinos, e alguns periféricos internos, como módulos PWM, ver apêndice A.

[5] Figura 5:
Disponível em: http://www.robix.com/contents.html.
Acesso em Set. 2016

- *Pulse Width Modulation* (PWM): Também chamada de Modulação de Largura de Pulso no português, tal técnica é empregada em diversas áreas da eletrônica, e consiste em alterar a largura do pulso de uma onda quadrada com o objetivo de controlar a potência ou velocidade. O período da onda quadrada é mantido, apenas a largura dos pulsos é alterada. Na tabela 1 do apêndice A, pode-se observar que o microcontrolador detalhado possui dois módulos (periféricos) específicos para PWM.

  A figura 6 apresenta alguns exemplos de forma de onda, variando a largura do pulso. O termo *Duty Cycle* que aparece nos exemplos é uma nomenclatura comum da técnica PWM, usada para referenciar a largura do pulso em nível ON (nível alto):

Figura 6 – Exemplos PWM [6]



Fonte: http://www.mecaweb.com.br

  Todos os exemplos apresentados acima partem de uma mesma onda, pode se observar, que em todos os casos o período da onda é mantido, apenas o *Duty Cycle*, é alterado.

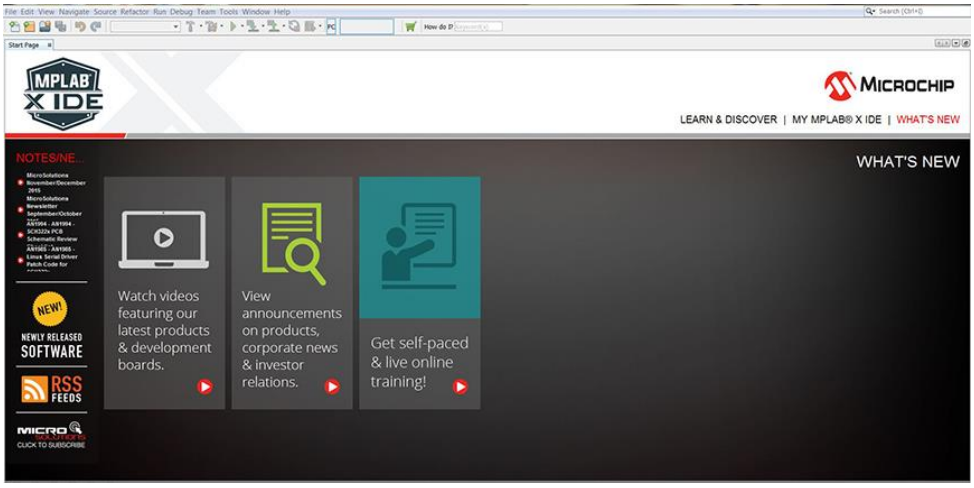
- Temporizadores: Além do periférico PWM, o PIC18F4550 possui temporizadores, que é basicamente um registrador que opera como contador ou temporizador; o circuito específico que faz a contagem e a temporização é controlado pelos bits do registrador temporizador e outros registradores especiais.

3.1.6 MPLABx

É um ambiente de desenvolvimento integrado (IDE) para programação, simulação e gravação de microcontroladores. É fornecida gratuitamente pela *Microchip Technology*. O mesmo conta com alguns compiladores embutidos, e tem suporte para a instalação de outros. A figura 7 ilustra a interface do software.

[6] Figura 6:
Disponível em: http://www.mecaweb.com.br/eletronica/content/e_pwm.
Acesso em Out. 2016

Figura 7 – MPLAx [7]



Fonte: http://www.microchip.com

- Compilador MPLAB C18 (MC18): É um compilador projetado para gerar executáveis para microcontroladores da linha PIC18, com suporte ao padrão ANSIC, porém otimizado. Uma das vantagens do MC18 é que o mesmo conta com uma biblioteca para manipulação de hardware, incluindo PWM, SPI, I2C, UART, manipulação de string e funções matemáticas; o que facilita o desenvolvimento.

3.1.7 Python

Python é uma linguagem interpretada e interativa de alto nível, de tipagem dinâmica e forte; e orientada a objetos. Sendo amplamente respeitada pela comunidade de Software Livre, possui muitas fontes de pesquisas seguras, com suporte bem definido e extenso.

O *Python* compila e funciona em praticamente todas as principais plataformas, como windows e linux, devido sua implementação ser escrita em ANSIC, que incorpora módulos, exceções, tipos de dados e classe de alto nível (JR et al., 2015).

- Django: É um framework web de alto nível, que estimula o desenvolvimento rápido, limpo, e sem repetições corriqueiras de código. É construído sobre o Python. Algumas características do Django serão apresentadas abaixo (JR et al., 2015):

  1. Mapeamento Objeto-Relacional (ORM): Com o ORM do Django você define a modelagem de dados através de classes em Python. Com isso é possível gerar suas tabelas no banco de dados e manipulá-las sem necessidade de utilizar SQL (o que também é possível).
  2. Interface Administrativa: No Django é possivel gerar automaticamente uma interface para administração para os modelos criados através do ORM.

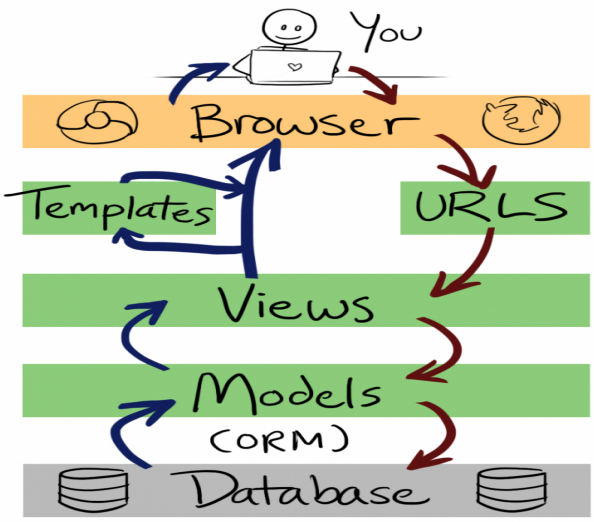
[7] Figura 7:
Disponível em: http://www.microchip.com/images/default-source/development-tools/mplab-x—3.jpg.
Acesso em Out. 2016

3. Formulários: É possível gerar formulários automaticamente através dos modelos de dados.
4. URL's Elegantes: No Django não há limitações para criação de URL's elegantes e de maneira simples.
5. Sistema de Templates: O Django tem uma linguagem de templates poderosa, extensível e amigável. Com ela você pode separar design, conteúdo e código em Python.
6. Sistema de Cache: O Django possui um sistema de cache que se integra ao memcached ou em outros frameworks de cache.
7. Internacionalização: Django tem total suporte para aplicações multi-idioma, deixando especificar strings de tradução e fornecendo ganchos para funcionalidades específicas do idioma.

- PYUSB: É um módulo Python para comunicação USB. Possui algumas rotinas para o envio e recebimento de dados.
- *Design Pattern MVC* (Model, View, Controller): Django é baseado no design pattern MVC, mas com suas particularidades. Veja abaixo (JR et al., 2015):
  - Model (M): são classes empregadas para interagir com o banco de dados, por isso é composta pelo ORM (Object Relational Mapping).
  - Controller (C): são classes responsáveis pela lógica de aplicação e o envio de requisições e respostas. No Django esta tarefa é realizada por um arquivo chamado *views.py* em conjunto com o arquivo de mapeamento de requisições *urls.py*, que representam o controller. Nesta camada, ainda estão todos os outros mecanismos entre a requisição e a resposta, como: handler, middlewares e URL dispatcher.
  - View (V): é a camada que apresenta dados e interage com o mundo exterior. É o que o usuário vê. Esta tarefa é executada por um sistema de Templates(T) no django.

  Por estes motivos o Django possui um modelo interno equivalente chamado MTV, realizando as mesmas tarefas do conhecido design pattern MVC. Veja a figura 8.

Figura 8 – *Design Pattern* MTV [8]



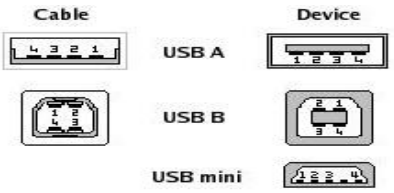
font: http://blog.easylearning.guru

3.1.8 Universal Serial Bus (USB)

O protocolo de comunicação USB foi desenvolvido por um conjunto de empresas que identificaram que a diversidade de protocolos e conectores existentes estavam trazendo, cada vez mais, dificuldade na comunicação de dispositivos com os computadores. Para a solução desse problema resolveram então criar um protocolo universal, definindo também seu respectivo conector. A finalidade era definir um padrão para ser implementado em qualquer dispositivo, com configuração fácil por parte do usuário, de baixo custo de desenvolvimento, ser Plug and Play e ter disponível vasta documentação e suporte. A velocidade da transmissão de dados também foi algo em foco nesse desenvolvimento. (SANTOS, 2009). Atualmente existem algumas versões desse protocolo, a mais atualizada é a 3.1, porém iremos utilizar a versão 2.0 como referência nesse trabalho. A figura 9 ilustra os principais tipos de conectores USB.

[8] Figura 8:
Disponível em: http://blog.easylearning.guru/wordpress/wp-content/uploads/2015/08/Django-Template.png.
Acesso em Out. 2016

Figura 9 – Conectores USB [9]



Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

Fonte: http://www.instructables.com

1. **Fluxo de dados**

   Os dispositivos USB são usados em diversas aplicações. Para essas podemos definir qual o fluxo de dados mais viável para satisfazer a demanda em específico. Segundo usb.org (2017), o protocolo USB define 4 tipos de fluxo de dados. São eles:

   - **Transferência de Controle:** Usada para o reconhecimento e configuração do dispositivo. Acontece à troca de dados de informações sobre o dispositivo, disponibilizando os descritores e parâmetros de configuração. Nesse tipo de transmissão há um robusto controle de erros, pois a troca de dados deve ser precisa.
   - **Transferência de Massa:** É o tipo de transferência que recebe e envia grande quantidade de dados, podendo existir simultaneamente o envio e o recebimento. Geralmente permanecem com uma conexão por muito tempo e variam a velocidade de transmissão de acordo com a carga de congestionamento do transmissor e do receptor. Geralmente usada para a transmissão de arquivos, pois provê forte controle de erros, assim como a transferência de controle. Os dispositivos de armazenamento como pen drivers, cartões de memória, impressoras, scanners entre outros usam este tipo de transferência.
   - **Transferência de Interrupção:** Usada para enviar pequena quantidade de dados. Geralmente são caracteres ou coordenadas de dispositivos de interface humana como mouses, teclados, controladores de jogos e outros. Os dados são gerados a partir de uma detecção de mudança no estado do dispositivo que é chamada de evento. Como por exemplo, o pressionar de um botão em um controlador de jogo ou movimentar o mouse (SANTOS, 2009).
   - **Transferência Isossíncrona:** Transferência isossíncronos são trocas de dados contínuas, geralmente em tempo real. Este tipo de transferência deve ser efetuado de acordo com uma determinada taxa previamente configurada. O envio de dados

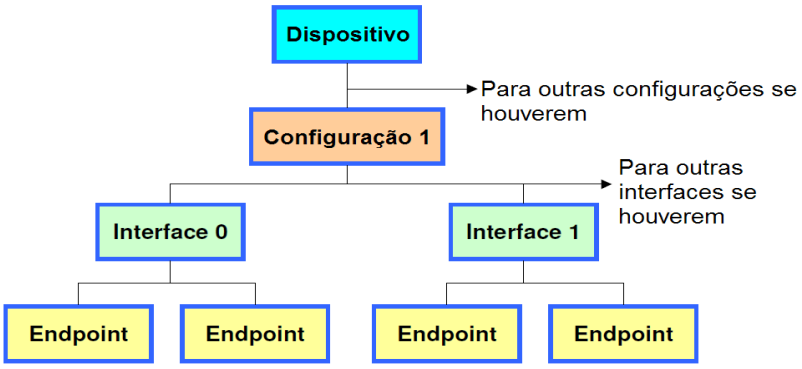
[9] Figura 9:
Disponível em: http://www.instructables.com/id/Male-to-Male-A-to-A-USB-Cable/
Acesso em Julho. 2017

é feito de acordo com essa taxa, desprezando eventuais perdas de blocos de dados neste período. Não há uma preocupação com a perda de dados, mas sim com a taxa de transmissão. Um exemplo de uso deste tipo de transferência é o uso de dispositivos de comunicação de áudio em tempo real ou de telefonia. Devem seguir uma taxa específica; caso exista alguma perda de dados no meio da transmissão não há sentido a retransmissão desses dados, pois não é mais interessante. Não há uma rigidez quanto à correção de erros, contudo há uma preferência quanto à taxa de transmissão. Geralmente neste tipo de transmissão o dispositivo faz uma reserva de memória para o recebimento dos dados no inicio da transmissão.

2. **Descritores**

   Todos os dispositivos USB têm uma hierarquia de descritores que informam ao host as características dos dispositivos, como: produtoID, vendorID, identificação do fabricante, tipo do dispositivo, número de configurações, número de endpoint (local físico onde será armazenado o fluxo de dados), tipo de transferência, tipo de interface, entre outros (USB.ORG, 2017). A figura 10 ilustra a hierarquia desses descritores.

Figura 10 – Hierarquia de descritores USB [10]



Fonte: http://tcc.ecomp.poli.br

3. **Classes de dispositivos**

   Os dispositivos USB podem ser divididos em classes, tais especificam basicamente o tipo de dispositivo, sua hierarquia de descritores e seu tipo de fluxo de dados (USB.ORG, 2017). Segue abaixo os tipos de classes mais comuns:

   - **Human Interface Device (HID)**

     Fazem parte desta classe dispositivos que geralmente ajudam os operadores a se comunicar de maneira fácil com os computadores. Sua instalação é simples, pois já estão previamente definidos os protocolos de comunicação para estes dispositivos. São dispositivos HID os mouses, teclados, controladores de jogos, etc. Geralmente

[10] Figura 10:
Disponível em: http://tcc.ecomp.poli.br/20091/TCC%20-%20Leonardo%20Santos.pdf
Acesso em Julho. 2017

a implementação destes dispositivos não necessita de muito trabalho, pois os protocolos de comunicação já são disponíveis prontos e já estão implementados nos sistemas operacionais.

- **Mass Storage Device (MSC)**

  Enquadram nesta classe dispositivos de armazenamento de grande quantidade de dados. Esses dispositivos são normalmente equipamentos que trabalham com transferência de um grande volume de dados, normalmente transferência de arquivos. São dispositivos que se identificam com essa classe os discos rígidos magnéticos externos, drives ópticos externos, incluindo os leitores e gravadores de CD e DVD, dispositivos portáteis de memória flash, câmeras digitais, entre outros.

- **Communications Device Class (CDC)**

  Os dispositivos que enquadram a classe CDC implementam um mecanismo de comunicação de propósito geral que pode ser usado para a comunicação entre a maioria dos dispositivos. Os dispositivos classificados como CDC são modems, dispositivos de rede, comunicação sem fio, telefonia.

3.2 Fundamentos históricos

Essa seção reúne trabalhos que versam sobre robótica móvel e educativa que impliquem direta ou indiretamente no conteúdo desse trabalho.

3.2.1 Robótica Educacional

Nessa subseção serão explicitados trabalhos que versam em sua maioria, sobre Robótica Educacional. Tais trabalhos serão subdivididos em sub tópicos dessa.

1. **A Robótica Educacional No Ensino Fundamental: Perspectivavas e Prática:** Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina. O trabalho (ZILLI, 2004) expõe como objetivo geral uma análise do uso da Robótica Educacional como recurso pedagógico, apontando algumas formas de uso da robótica nas escolas de Curitiba (PR), avaliando as perspectivas em relação ao processo cognitivo. Destaca, ainda, a importância do uso da tecnologia na educação, apresentando as visões de alguns autores consagrados da atualidade, como Gardner, Perrenoud, Papert e Piaget. E por fim, os resultados obtidos em uma pesquisa realizada nas escolas de ensino fundamental de 5ª a 8ª séries de Curitiba (PR) que utilizam a Robótica Educacional como recurso. A partir destes resultados, é feita uma proposta de implantação da tecnologia em questão nas escolas, dando ênfase na metodologia de aplicação desta proposta de ensino. Por fim, Zilli (2004) concluiu que sem dúvida nenhuma, a Robótica Educacional é uma alternativa interessante como ferramenta pedagógica no processo ensino-aprendizagem, e que os kits robóticos educacionais no mercado são importantes para isso, principalmente os da empresa *Lego*, pois segundo ela, sua metodologia facilita, justamente por ser sólida e bem definida, colaborando com o professor incluído no processo. Adicionalmente, foi

realizada uma proposta de implantação da Robótica Educacional, com base nos dados levantados na pesquisa.

2. **RoboEduc: Uma Metodologia de Aprendizado com Robótica Educacional:** Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Norte (UFRN). O trabalho (SILVA, 2009) propôs uma metodologia para o ensino de robótica no ensino fundamental da educação básica no Brasil, baseada na teoria sócio-histórica de *Lev Vygotsky*, um renomado professor de literatura, pioneiro no conceito de que o desenvolvimento intelectual das crianças. Esta metodologia em conjunto com o kit Lego® Mindstorms® e um software educacional compõem o sistema de robótica pedagógica denominado RoboEduc. Foi realizada uma oficina com algumas atividades de robótica para a participação de crianças com idade entre 8 a 10 anos que visaram a produzir conhecimento sobre a construção, programação e controle de protótipos robóticos, o que foi fundamental para a análise da robótica como mediadora de ensino-aprendizagem. Por fim, (SILVA, 2009) concluiu, propondo uma metodologia para a robótica educativa, sendo que a mesma afirmou ser uma das principais contribuições deste trabalho. Tal metodologia foi validada pelos pressupostos da teoria de *Vygotsky*.

3. **O Uso da Robótica Educacional no Ensino Fundamental: relatos de um experimento:** Monografia apresentada ao curso de Bacharelado em Ciência da Computação da Universidade Federal de Goiás. O trabalho (PEREIRA, 2010) retrata a progressiva queda de interesse por cursos ligados a informática, seja de nível de graduação ou de ensino médio, por parte dos jovens. Um estudo sobre robótica educativa foi realizado, e aplicado a jovens da zona rural de uma cidade, a iniciativa foi incorporada a um projeto, que recebeu o nome de *Levando a Informática do Campus ao Campo*. O objetivo geral do projeto é apresentar conceitos de computação e robótica educacional aos alunos do ensino fundamental da zona rural do município de Catalão - GO. O projeto foi ministrado em módulos, no período das férias escolares. Por fim, Pereira (2010) concluiu que é prematuro afirmar que os alunos submetidos ao projeto *Levando a Informática do Campus ao Campo*, foram definitivamente atraídos para cursar cursos de Computação, porém as expectativas são boas. Foi observado que o uso do software Lego® Mindstorms® NXT de fato auxilia e facilita o aprendizado de programação.

   Os trabalhos acima apresentados possuem em comum uma fundamentação educacional. Buscando desenvolver propostas de implantação, metodologias de uso e estudos de caso de robótica educativa. Da mesma forma o presente trabalho possui um ideal educacional, com o intuito de que os produtos nele desenvolvidos sejam utilizados futuramente para ensino nas escolas.

3.2.2 Robótica Móvel

Nessa subseção serão explicitados trabalhos que versam em sua maioria, sobre Robótica Móvel. Tais trabalhos serão subdivididos em sub tópicos desse.

1. **Protótipo de um Robô Móvel de Baixo Custo Para Uso Educacional:** Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá. O trabalho (GONÇALVES, 2007) propôs o desenvolvimento alternativo para a construção de projetos na área de robótica educacional. Tais são de baixo custo e de fácil implementação, acessíveis à realidade de muitas escolas brasileiras. A placa controladora para a construção dos projetos recebeu o nome de *GoGo*, e foi implementada no decorrer do trabalho. Em resumo, segundo Gonçalves (2007), a construção do robô móvel baseou-se na placa *GoGo*, e alguns componentes de baixo custo com reaproveitamento de sucata eletrônica.

2. **Construção de um Manipulador Robótico de Baixo Custo Para Ensino:** Monografia apresentada ao curso de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel. O trabalho (LAZZARIM, 2012) apresenta o desenvolvimento de um braço manipulador robótico, de baixo custo, voltado para ensino, juntamente com uma interface controladora. Segundo Lazzarim (2012), a conclusão do trabalho obteve êxito baseado na otimização do custo, o qual ficou abaixo da referência escolhida.

   Os trabalhos explicitados acima discorrem sobre robótica móvel, especificadamente as existentes em robôs de quatro rodas, e em braços robóticos. Da mesma forma, embora o presente trabalho possui a iniciativa de ser um produto inicial, posteriormente o mesmo poderá ser continuado e aprimorado para o controle efetivo de um braço robótico, considerando que propõe como resultados produtos que permeiam esse controle.

4 Interface Web Controladora

“O período de maior ganho em conhecimento e experiência
é o período mais difícil da vida”.
Dalai Lama

O desenvolvimento desse trabalho pode ser dividido em duas etapas, produção de um sistema web e produção de um protocolo de comunicação permitindo a comunicação USB entre o sistema e a controladora para manuseio do braço robótico.

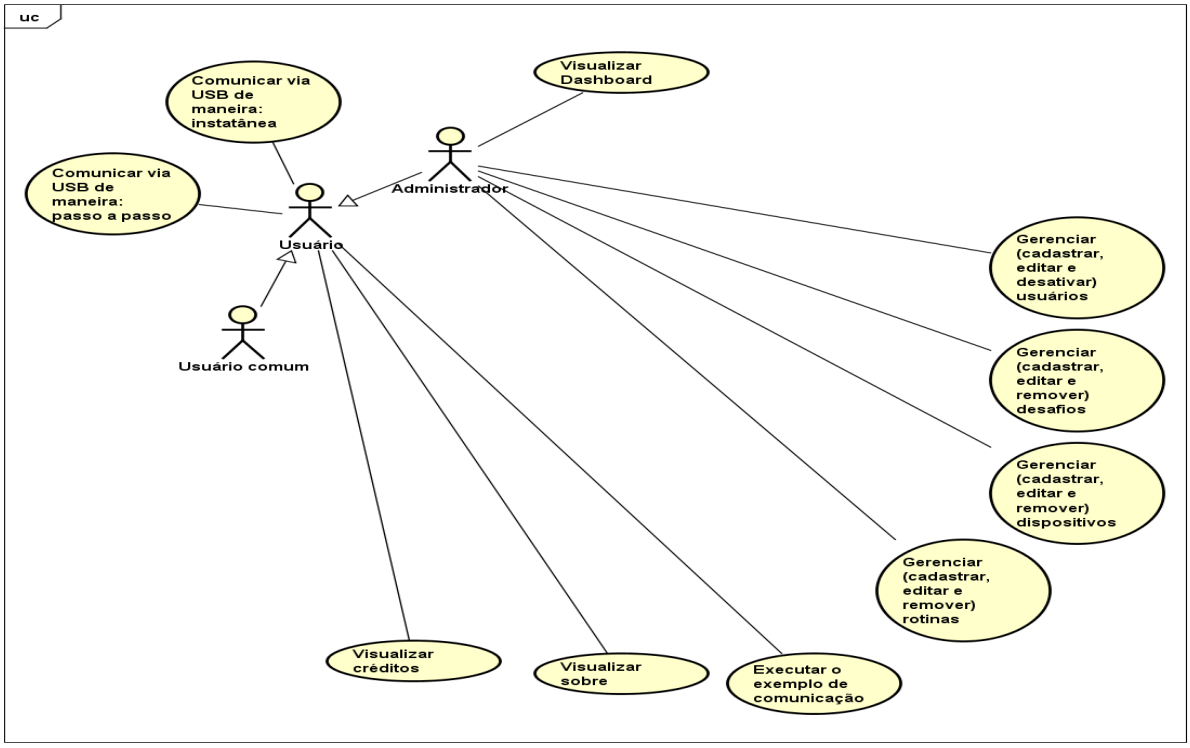
4.1 Sistema Web

Nesse tópico será detalhado o processo metodológico para o desenvolvimento do sistema. Durante o desenvolvimento do mesmo foram colocados em prática vários conhecimentos adquiridos no decorrer da graduação, também foram aprendidos e utilizados tecnologias atuais. Estruturado sobre plataforma web, seu uso só será possível se houver um navegador instalado na máquina do usuário. A arquitetura e a escrita foi criada em Python. Estruturado em camadas no modelo Model-View-Template (MVT) com o uso do framework Django, principal biblioteca utilizada no desenvolvimento, o mesmo auxiliou e facilitou a se chegar à solução de maneira rápida e organizada. O banco de dados utilizado foi o SQLite, devido ao Django ter suporte ao mesmo. Mais detalhes descritos nos próximos subtópicos. Para maior facilidade de se referenciar esse sistema, foi dado um nome ao mesmo, ROPIC Admin.

4.1.1 Processo de desenvolvimento do software

A seguinte sequência define como ocorreu o processo de desenvolvimento. A princípio foram levantados os requisitos, buscou-se traçar um objetivo geral, todas as funcionalidades do sistema foram escritas, a fim de iniciar a resolução do problema. Uma análise desses requisitos foi realizada em seguida, nesse ponto todos os requisitos começaram a ser construídos como parte do produto final, foi realizado um esboço das classes e por fim uma modelagem visando o caminho a ser trilhado. Em seguida definiu-se a arquitetura do sistema, a linguagem de programação de escrita, o sistema de gerenciamento de banco de dados a ser utilizado, o padrão de interface gráfica, etc. Após essas etapas preliminares deu se início a implementação. A figura 11 ilustra o diagrama de casos de uso do sistema.

Figura 11 – Diagrama de casos de uso



Fonte: elaborado pelo autor

4.1.2 Tecnologias

Definir as tecnologias a se utilizar em um trabalho como esse, talvez seja uma das decisões mais importantes, a má escolha pode estender o tempo de desenvolvimento, trazer problemas, e o pior, fazer com que o desenvolvedor não chegue à solução final. Para a definição das tecnologias a serem utilizadas, levou-se em conta duas diretrizes:

1. O desenvolvimento será simples, rápido e de qualidade.
2. A melhor opção para se chegar à solução, não importando se o autor tem ou não domínio do uso de tal tecnologia.

Obedecendo tais diretrizes escolheu-se o framework de alto nível python web, o Django. Suas principais vantagens são: simples, gratuito de código aberto, seguro, documentando de forma organizada e progressiva, implementa Mapeamento Objeto-Relacional (ORM), implementa o conceito de URLs Amigáveis e o conceito de Template Extending, conceito em que se estende o vocabulário de criação de uma página html (DJANGOPROJECT.COM, 2017).

O banco de dados SQLite foi o que melhor se encaixou às diretrizes. O mesmo executa scripts SQL. Seu funcionamento é por meio de leitura e escrita em arquivo no disco, e suas principais vantagens são: multiplataforma, fácil utilização e manipulação, gratuito e de código aberto. Ao contrário da maioria dos bancos de dados SQL, o SQLite não executa um processo

separado no servidor com um sistema de gerenciamento de banco de dados (SGBD). Estima-se que o mesmo é o banco de dados mais implantado no mundo (SQLITE.ORG, 2017).

Além do python ser um ótima linguagem de se programar, o motivo principal do autor ter escolhido tal linguagem, é pelo fato da mesma possuir um módulo com implementação de uma biblioteca gratuita, de código aberto, que oferece comunicação entre dispositivos USB de maneira simples e fácil, a Pyusb (WALAC.GITHUB.IO, 2017). Atualmente, é a biblioteca mais simples, disponível, de se implementar uma rotina de comunicação USB.

O framework AngularJS foi escolhido e utilizado para um controle maior de quais componentes HTML dos templates precisavam ser renderizados ou não na tela do navegador, evitando a perda de desempenho do mesmo. O AngularJS é um framework JavaScript, que permite a expansão da sintaxe do HTML para expressar os componentes da aplicação de forma clara e sucinta (ANGULARJS.ORG, 2017).

Os frameworks frontend utilizados foram o Boostrap e o Semantic UI. Ambos frameworks gratuitos, com muitos componentes gráficos prontos. O uso desses deram um elegância a mais ao projeto.

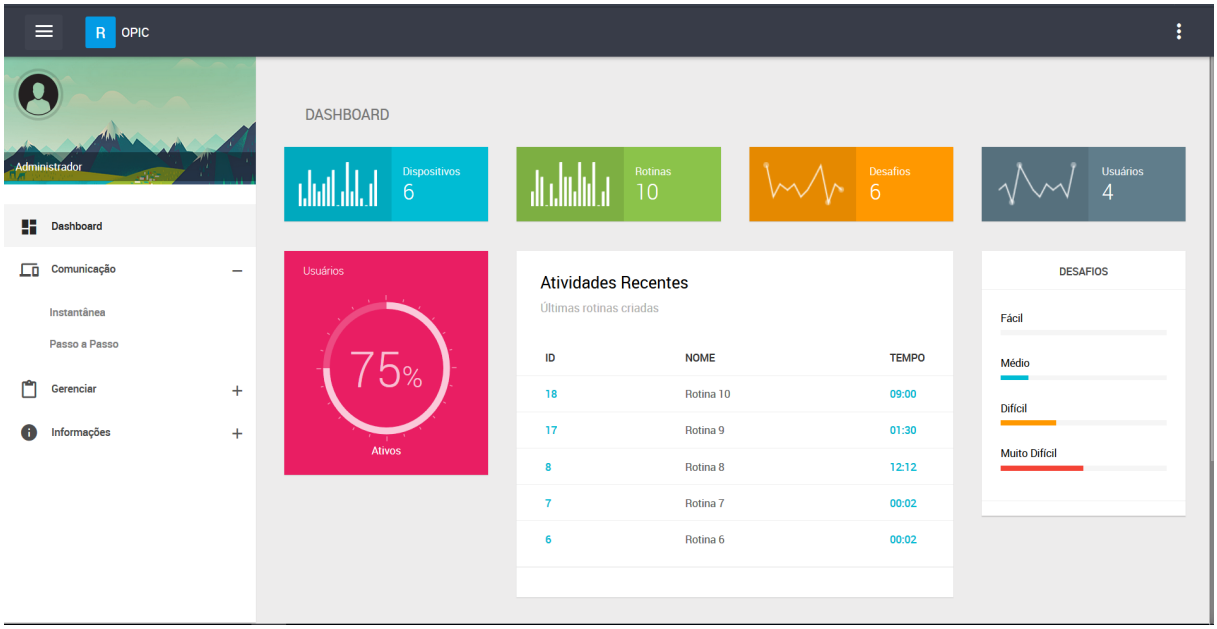
4.1.3 Ferramentas e configuração do projeto

A ferramenta Visual Studio 2017 Community, uma versão gratuita para estudantes, foi utilizada como ambiente integrado de desenvolvimento (IDE). Tal ferramenta oferece a opção de se criar um projeto Django inicial pré configurado. É necessário que o usuário tenha Python versão 3.4.4 para que os módulos do projeto sejam instalados corretamente. Por ser uma ferramenta profissional, o uso da mesma cooperou em produtividade (VISUALSTUDIO.COM, 2017).

4.1.4 Resultado

Como resultado foi obtido um sistema web com interação com o usuário, o mesmo é preparado para uma controladora de braço robótico, podendo ser aperfeiçoado com intuito de promover mais opções e funcionalidades para o uso do usuário. O sistema é composto basicamente por uma tela inicial, dando a opção do usuário começar. Ao começar o mesmo será redirecionado a tela de login, e ao efetuar o login, redirecionado ao *dashboard*, um painel de informações rápidas, figura 12. A partir daí o usuário poderá gerenciar (cadastrar, editar, remover e visualizar) rotinas, desafios, dispositivos e usuários, ou até mesmo iniciar a comunicação em algum dos modos existentes, comunicação instantânea que consistirá em controlar o braço robótico pelo teclado do computador escolhendo os motores e incrementando ou decrementando os ângulos dos mesmos, ou comunicação passo a passo que consistirá em controlar o braço robótico através de execução de rotinas. Vale ressaltar que o usuário terá a opção de executar o exemplo descrito no tópico 7 da subseção 4.2.1.

Figura 12 – Sistema - Resultado

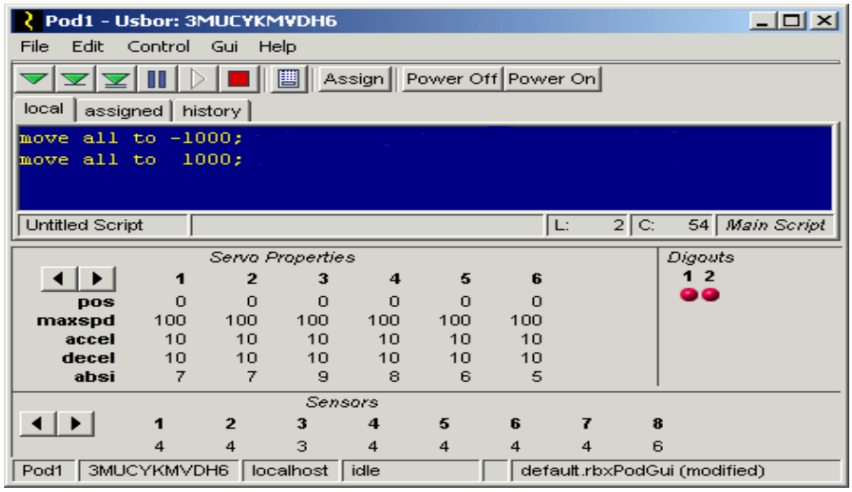


Fonte: elaborado pelo autor

4.1.5 Limitações

Para a identificação das limitações desse sistema proposto, foi realizada uma abordagem simples aos comandos de movimento do braço permitidos pelo mesmo, comparando-os com os do sistema original (chamado de Usbor) do braço robótico Robix Rascal. Foi realizada uma análise completa aos comandos de movimento do robix, porém para esse trabalho apenas dois desses comandos são importantes, "Move" e "Macro", o primeiro é para movimento dos servos motores existentes no braço, e o segundo uma espécie de bloco de comandos que pode ser chamado em qualquer parte do programa e que internamente pode conter inúmeros comandos "Move". Um exemplo de comandos pode ser visto na figura 13.

Figura 13 – Tela do programa Robix Rascal



Fonte: Adaptado de *print screen* da tela do programa Usbor.

Nesse *script* de comandos descrito na figura 13, todos os motores estão sendo movidos para a posição -1000 e em seguida para a posição 1000. Breve explicação no bloco de código abaixo, com exemplos de uso dos comandos tidos como importantes acima.

```
#-----------------COMANDO MOVE-----------------#

move <servo list> to <posval> | by <val>

#____________________EXEMPLOS____________________#

    move 1,2,3 to 0;
    # (Move os motores 1,2,3 para 0)

    move 3,4 to -300, 1 to maxpos, 5 by -250
   # (Move os motores 3,4 para -300,
   #  o motor 1 para a posição máxima
   #  e o motor 5 diminui sua posição em 250 unidades)

#-----------------COMANDO MACRO-----------------#
    macro <macro name>
        <command>; <command>; <command>
    end

#____________________EXEMPLOS____________________#
    macro TCCII
        move 1 to 0;
    end
    # (Move o motor 1 para 0)]

    TCCII
    # (chama o macro, o bloco será executado 1 vez)

    TCCII 1
    # (chama o macro, o bloco será executado 1 vez)

    TCCII 2
    # (chama o macro, o bloco será executado 2 vezes)

    TCCII 0
    # (chama o macro, o bloco será executado infinitamente)
```

O sistema desenvolvido nesse trabalho possui uma proposta diferente com relação ao Usbor. O mesmo foi preparado apenas para permitir movimentos simples, equivalentes aos movimentos "Moves", mas não complexos como a execução de comandos "Macro". Porém, nada impede que mais comandos sejam criados e os que existem aprimorados, para que o sistema desenvolvido os execute de maneira equivalente ao Usbor. Depois da análise e da identificação dessas limitações, foi possível identificar que a implementação desenvolvida nesse trabalho é eficaz quanto a realização de rotinas de movimentos básicos.

Vale ressaltar que nessa implementação foram criados dois protocolos de comunicação não existentes no Usbor, tais serão descritos na próxima subseção.

4.1.6 Protocolo de comunicação

No presente trabalho foram desenvolvidos dois protocolos de comunicação, instantânea e passo a passo. Em ambos os casos o usuário deverá analisar as explicações iniciais, em seguida conectar ao dispositivo robótico (braço), configurar o dispositivo (informando basicamente o número de motores do braço) e por fim começar a comunicação.

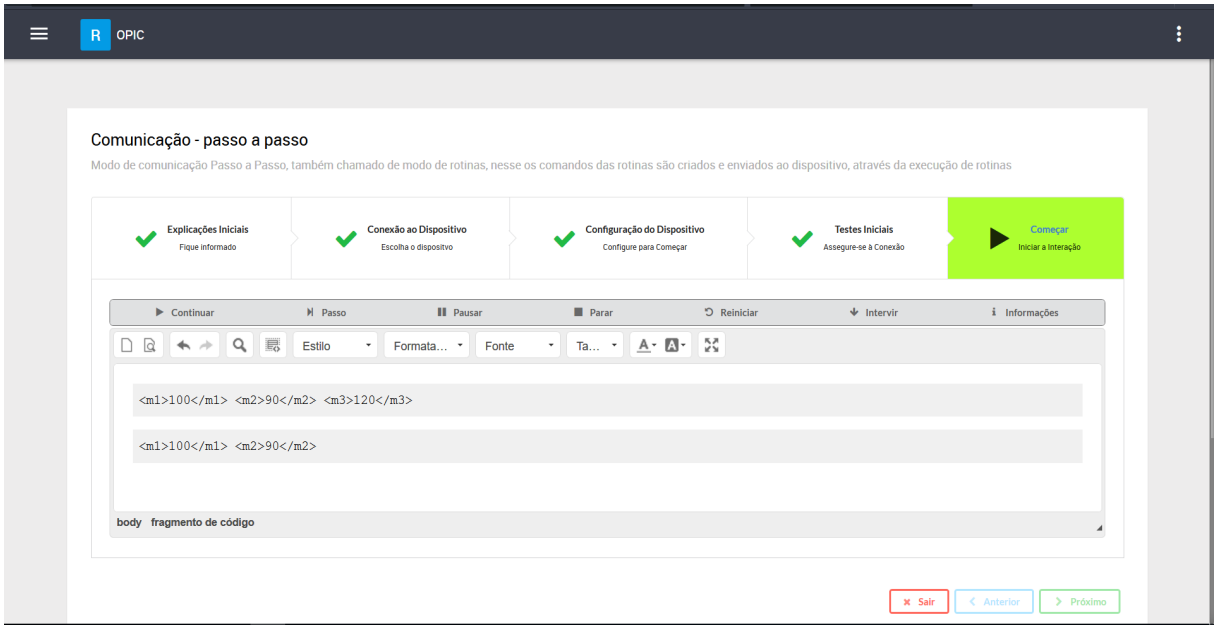
No modo instantâneo, figura 14, o usuário movimentará o braço através de comandos enviados pelo teclado, os numerais de 1 a 8 serão usados para selecionar os motores, as setas *left* e *right* para decrementar e incrementar respectivamente, os ângulos dos motores – motor 1 está selecionado, motivo pelo qual o mesmo é apresentado em borda escura.

Figura 14 – Tela de comunicação instantânea

Fonte: elaborado pelo autor

No modo passo a passo, figura 15, o usuário movimentará o braço através de execução de rotinas, o mesmo incluirá um bloco de código em que conterá os comandos para os movimentos desejados. A notação dos comandos segue a notação HTML – observar dois blocos de código, o primeiro "**<m1>100</m1> <m2>90</m2> <m3>120</m3>**", indicando que o motor 1 vai para a posição 100, o motor 2 para a 90 e o motor 3 para a 120. O mesmo acontece no segundo bloco, porém não existe movimento no motor 3.

Figura 15 – Tela de comunicação passo a passo



Fonte: elaborado pelo autor

Para enviar os dados à controladora que poderá ser desenvolvida, é sugerido que primeiramente seja enviado um vetor com os dados de configuração, informando o modo de comunicação, a quantidade de motores, e só em seguida seja enviado um outro vetor de informações de movimentos. Como esse vetor possui 64 Bytes (tópico 5, subseção 4.2.1) disponíveis para envio de dados, é recomendável que cada posição seja para um motor, conforme a figura 16 ilustra.

Figura 16 – Exemplo de envio de dados pelo protocolo de comunicação

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	.....	Byte 64
Motor 1	Motor 2	Motor 3	.....	.....	.....	.....	.....	.....	.....
90º	180º								
0101 1010	1011 0100	.....	.....	.....	.....	.....	.....	.....	.....

<m1> 90 </m1>	<m2> 180 </m2>

Fonte: elaborado pelo autor

4.2 Comunicação USB - Circuito e Programa Embarcável

Nesse tópico será abordado e detalhado o processo de desenvolvimento de um programa para comunicação USB embarcável ao microcontrolador de 8 bits, PIC18F4550 da fabricante Microchip. Será detalhado também o circuito necessário para essa comunicação, que permitirá a interação entre o mesmo e o sistema web por meio da USB, as tecnologias e as ferramentas utilizadas.

4.2.1 Processo de desenvolvimento do programa

Primeiramente foi definido que a velocidade da transmissão seria de 12 Mbps (full speed), a maior suportada pelo microcontrolador, que a transferência seria por meio de interrupção e que o dispositivo USB seria da classe Human Interface Device (HID). A linguagem escolhida foi a linguagem C. Para compilar o programa foi escolhido o compilador MPLAB® C18, vale ressaltar que ao contrário de alguns outros compiladores pagos, o C18 não oferece e nem documenta uma biblioteca exclusiva para comunicação USB, ou seja, todas as rotinas USB escritas para o compilador C18 deverão ser criadas do princípio. A fabricante disponibiliza apenas demonstrações de comunicação USB para alguns de seus dispositivos, mas nada em específico para o chip escolhido. A codificação do programa foi norteada com a ajuda dessas demonstrações. Abaixo serão descritos os passos seguidos durante o desenvolvimento.

1. **Definição de variáveis de estado**: Para o controle da comunicação, foram definidos 7 estados possíveis para o dispositivo USB, a nomenclatura dos estados seguiu o padrão descrito no *datasheet* do chip utilizado.

    - **DETACHED_STATE**: Estado inicial do dispositivo. Nesse, o barramento, as interrupções e o módulo USB são desabilitados. É análogo a remoção do dispositivo USB da porta USB do computador, porém tal remoção é virtual e não é física.
    - **ATTACHED_STATE**: É o estado próximo ao estado DETACHED. Acontece basicamente o contrário do que foi feito no estado anterior. O barramento, as interrupções e o módulo USB são habilitados. É análogo ao plug do dispositivo USB na porta USB do computador.
    - **POWERED_STATE**: É o estado próximo ao estado ATTACHED. Nesse, o dispositivo está regulando as tensões nos canais de transferência de dados D+ e D-.
    - **DEFAULT_STATE**: É estado próximo ao estado POWERED. Pode-se dizer que é o estado de equilíbrio, estado padrão. Nesse, o barramento e alimentação do módulo está preparada para enviar e receber dados.
    - **ADR_PENDING_STATE**: Entre o estado DEFAULT e o ADDRESS foi definido um estado intermediário, que basicamente indica que o endereço do dispositivo USB está pendente, ainda não foi encontrado.
    - **ADDRESS_STATE**: É o penúltimo estado em que o dispositivo USB pode se encontrar. Nesse, o endereço do dispositivo USB foi encontrado, e o módulo USB do chip, já o possui.

- **CONFIGURED_STATE**: É o último estado em que o dispositivo USB pode se encontrar. Nesse, o dispositivo está preparado para enviar e receber dados.

2. **Configuração de clock**: O microcontrolador usado nesse trabalho possui três opções de configuração de fonte geradora de clock: oscilador primário, oscilador secundário e oscilador interno. Esses tipos podem assumir diversas configurações diferentes com cada tipo de oscilador ou a combinação entre eles, devido o hardware possuir um sistema adicional de escala (MICROCHIP, 2009), para maiores detalhes sobre a configuração de clock do chip, ver apêndice B.

3. **Configuração do cabeçalho**: Para começar a programação do microcontrolador deve-se definir a configuração do cabeçalho. Basicamente é essa configuração que diz como o chip vai se portar e quais de seus módulos serão utilizados. Para detalhes da configuração do chip para usar a USB, ver apêndice B.

4. **Construção dos descritores USB**: Para o computador reconhecer o dispositivo USB deve-se criar e configurar os descritores, que vão definir as características do dispositivo. Segundo Microchip (2009), existem oito diferentes tipos de descritores padrão, porém apenas cinco deles são mais importantes para se ter um dispositivo configurado. São eles, descritor do dispositivo, descritor de configuração, descritor de interface, descritor de endpoint e descritor de string. Detalhes das configurações de cada descritor, ver apêndice C.

5. **Definição de vetores de envio e recebimento de dados** : Foram definidos dois vetores de transferência de dados, um para envio e um outro para receber os dados, g_Buffer_EnviarDados e g_Buffer_ReceberDados, respectivamente. Ambos os vetores possuem 65 posições e cada posição é um byte. Atentar para não utilizar as posições g_Buffer_EnviarDados[0] e g_Buffer_ReceberDados[0], ambas são utilizadas pelo protocolo USB.

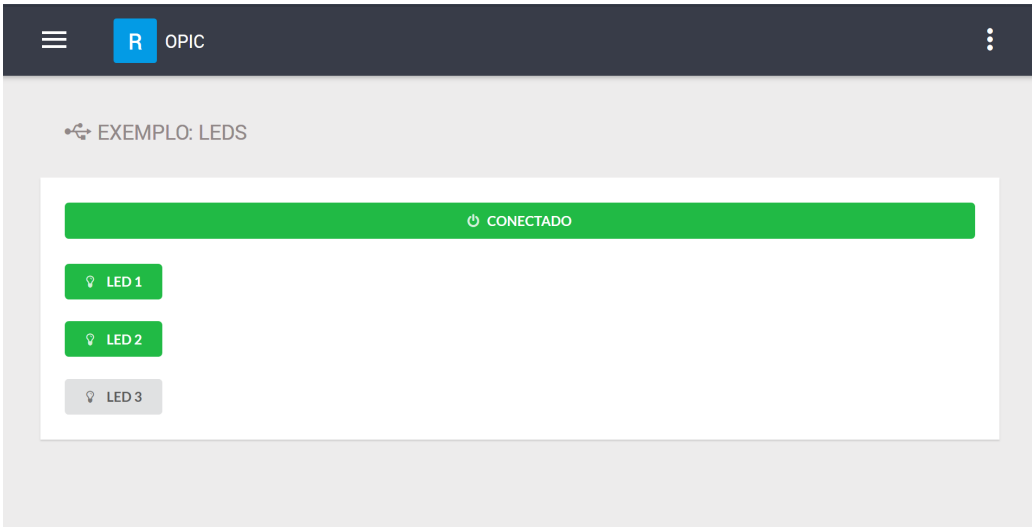
6. **Rotinas USB**: Para a comunicação USB foram definidas algumas rotinas com intenção de criar uma biblioteca. Esta biblioteca poderá ser utilizada em trabalhos futuros onde o desenvolvedor poderá apenas incluí-la ao projeto. Detalhes de cada rotina, ver apêndice D.

7. **Criação de um exemplo teste**: Para testar a comunicação USB foi criado um pequeno programa que basicamente envia a sequência de caracteres "ROPIC ADMIN." e recebe o nível lógico de três LEDs. Os vetores g_Buffer_EnviarDados e g_Buffer_ReceberDados, citados no tópico 5 dessa subseção, vão ficar com a seguinte configuração:

   - g_Buffer_EnviarDados[0:9]: Representa a sequência de caracteres.
   - g_Buffer_EnviarDados[1:3]: Representa o primeiro, o segundo e o terceiro LED, respectivamente.

   Após a criação da rotina teste o programa deve ser compilado. A compilação gera um arquivo no formato *.hex* (hexadecimal), arquivo que deverá ser gravado no chip e utilizado na simulação. A figura 17 ilustra o exemplo descrito no item 7 dessa subseção.

Figura 17 – Exemplo teste do envio de dados confirmados por leds



Fonte: elaborado pelo autor

4.2.2 Materiais

Os materiais utilizados para a montagem do circuito serão descritos nesse tópico. A escolha dos mesmos deu-se com a ideia de simplificar o circuito.

1. Itens de propriedade própria:
    - 01 Gravador PICKit3.
    - 01 Protoboard 16.5cm x 5.5cm.
    - 02 Capacitores de cerâmica de 220nF.
    - 21 Jumpers.
    - 01 Cabo Usb 1.5m.
2. Itens de propriedade do CEFET-MG Campus 7:
    - 01 Led Verde.
    - 03 Leds Vermelhos.
    - 04 Resistores de 460 Ω.
    - 01 Crystal Oscilador 20Mhz.
    - 02 Capacitores de cerâmica de 15pF.
    - 01 Microchip PIC18F4550.

Os laboratórios de circuitos elétricos e de embarcados do CEFET-MG Campus 7 foram utilizados durante o desenvolvimento do projeto, logo alguns de seus equipamentos foram usados.

4.2.3 Ferramentas e configuração do projeto

- MPLABx: IDE utilizada para desenvolver e debugar o programa de comunicação USB, veja a subseção 3.1.6
- Proteus: Simulador utilizado para a montagem e teste do circuito. Com o mesmo foi possível realizar os testes do programa de comunicação USB.

  O uso do Proteus foi essencial neste trabalho, para a execução de testes ao circuito, e a comunicação USB em si. Seu uso mais aprofundado permitiu um ganho a mais em conhecimento, realmente é ferramenta que instiga à aprendizagem. Basicamente seu uso permitiu a confecção do circuito da USB, a simulação de envio e recepção de dados via protocolo USB. A IDE MPLABx possui um *plugin* chamado Proteus VSM Viewer, através deste a simulação de ambos os softwares são interligadas, permitindo o *debug* de um software para o outro.
- Visual Studio Code: Editor de código usado para realizar testes à comunicação USB.

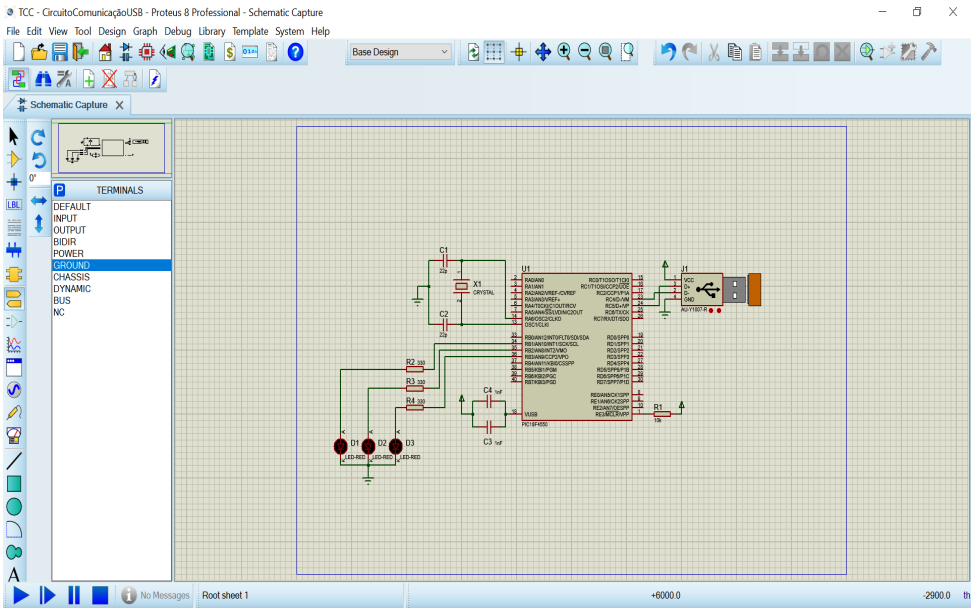
4.2.4 Circuito

O circuito foi confeccionado a princípio no simulador proteus, depois de testado, configurado e ajustado, partiu-se para a montagem física do mesmo em protoboard. No proteus foi possível desenvolver também o layout da placa de circuito impresso (PCB) em 2D e 3D. A ideia desse trabalho não foi confeccionar a placa de circuito impresso, mas mostrar o caminho a ser seguido para confecciona-la.

1. **Modelagem em simulador**: A modelagem no proteus foi realizada de maneira simples. Primeiro foram selecionados os componentes eletrônicos necessários, em seguida os valores desses componentes foram alterados. Um exemplo de alteração que ocorreu foi no resistor, o valor padrão do mesmo é 100 Ω, para o LED foi necessário altera-lo para 330 Ω. Os componentes utilizados foram 1 Microcontrolador Microchip PIC18F4550, 1 Oscilador Crystal 20Mhz, 1 Conector USB 2.0 tipo c, 3 Leds vermelhos, 3 Resistores de 330 Ω, 1 Resistor de 1k Ω, 2 Capacitores de cerâmica de 22pF e 2 Capacitores de cerâmica de 1nF.

   Vale ressaltar que os componentes acima não são equivalentes aos componentes utilizados no circuito físico, os quais foram descritos na subseção 4.2.2, indicando uma certa flexibilidade na montagem via simulador. Um exemplo disso é que para cumprir a obrigatoriedade com o intuito de se ter um bom funcionamento da USB no circuito físico, os capacitores de cerâmica de 22pF deveriam ser de 15pF (MICROCHIP, 2009). A figura 18 ilustra o circuito finalizado no simulador.

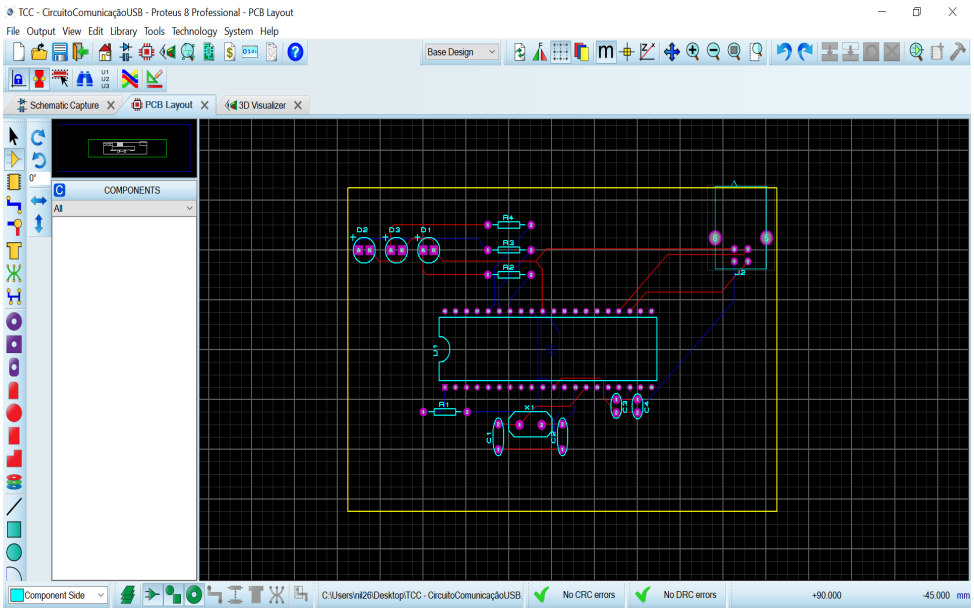
Figura 18 – Modelagem do circuito - Resultado final



Fonte: Adaptado de *print screen* da tela do Proteus.

Umas das grandes vantagens do proteus é a possibilidade de se modelar também o layout PCB do circuito. O escopo desse trabalho não inclui a confecção da PCB, o layout foi desenvolvido apenas por curiosidade, figura 19.

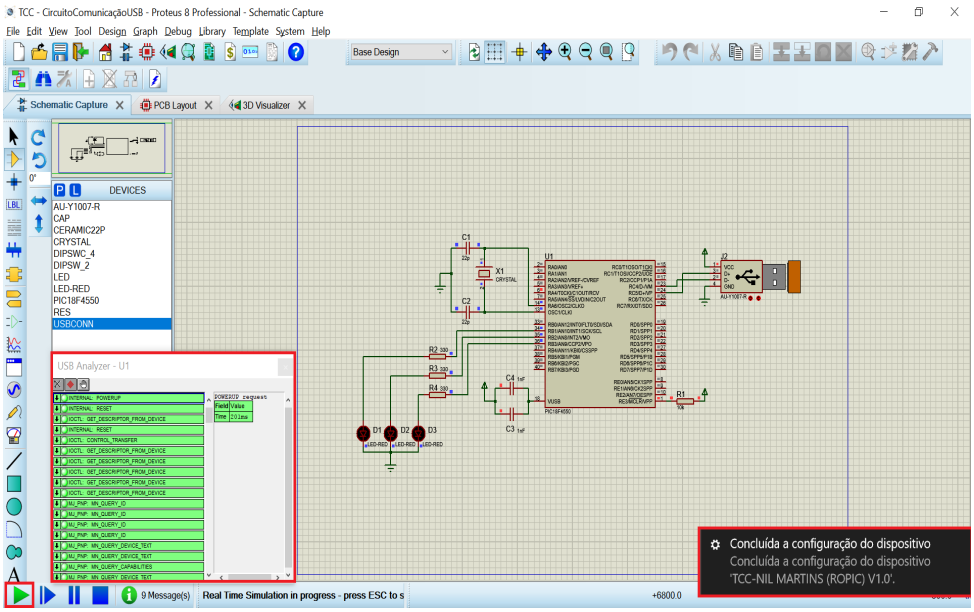
Figura 19 – Layout PCB - Resultado



Fonte: Adaptado de *print screen* da tela do Proteus.

2. **Testes em simulador**: Depois do circuito montado e configurado, iniciou os testes da USB, como ilustrado na figura 20. A principio é necessário selecionar o programa, o qual foi detalhado na subseção 4.2.1 e gerado no tópico 7 da mesma.

Figura 20 – Analisador USB

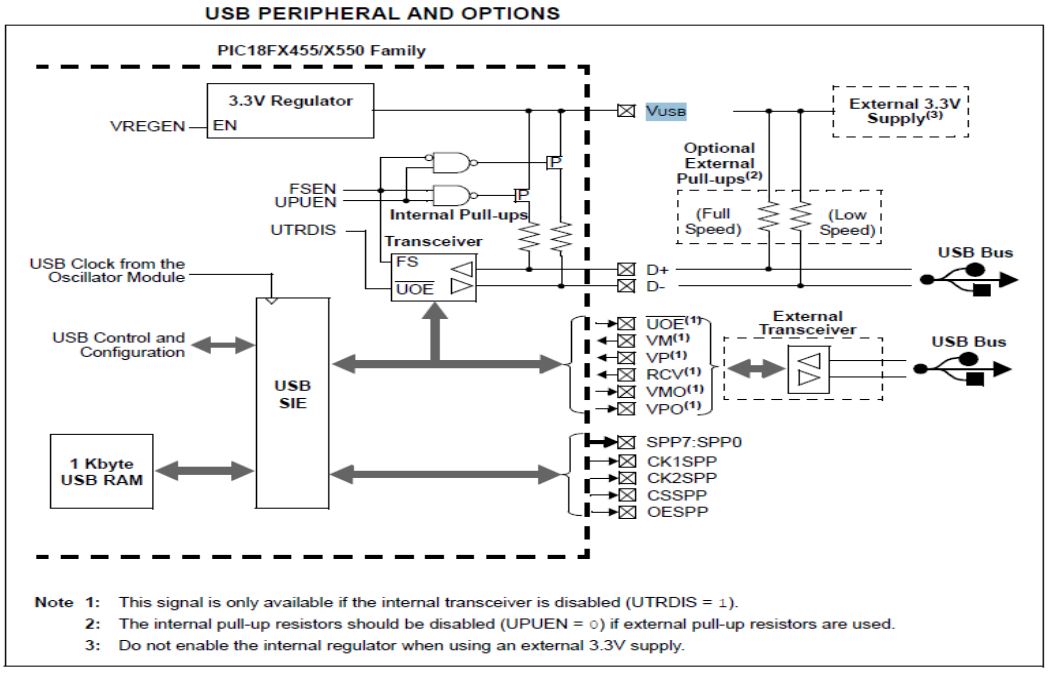


Fonte: Adaptado de *print screen* da tela do Proteus.

O simulador disponibiliza de uma opção que oferece a possibilidade do usuário instalar e agregar a ele um software que cria e simula virtualmente uma porta USB, fazendo com que os dispositivos USB simulados sejam "tratados" como dispositivos físicos, basicamente os mesmos são "plugados" nessa porta virtual quando a simulação é inicializada. Tal funcionalidade facilitou muito nos testes. Na figura acima podemos ver a simulação iniciada, ao iniciar a simulação o proteus abre uma janela (canto esquerdo da figura) que traz várias informações do dispositivo USB, como o status da transmissão, os dados que estão sendo transmitidos e recebidos, e todos os descritores USB, tudo isso em tempo real. No canto direito inferior pode-se visualizar que o sistema operacional realmente trata o dispositivo como físico.

3. **Modelagem física em protoboard**: Após o desenvolvimento do programa, e inúmeros testes em simulador, alguns ajustes no programa e no circuito que seria implementado fisicamente, iniciou a modelagem física, tendo como suporte o datasheet do chip. A primeira versão da modelagem apresentou erros e durante um bom tempo não se obteve a solução para o problema. O computador reconhecia que um dispositivo havia sido plugado, porém os descritores não eram recebidos por ele, não permitindo o sistema operacional (SO) realizar a configuração do dispositivo. A figura 21 ilustra o diagrama do hardware USB do microcontrolador, a análise do mesmo foi crucial para a solução do problema.

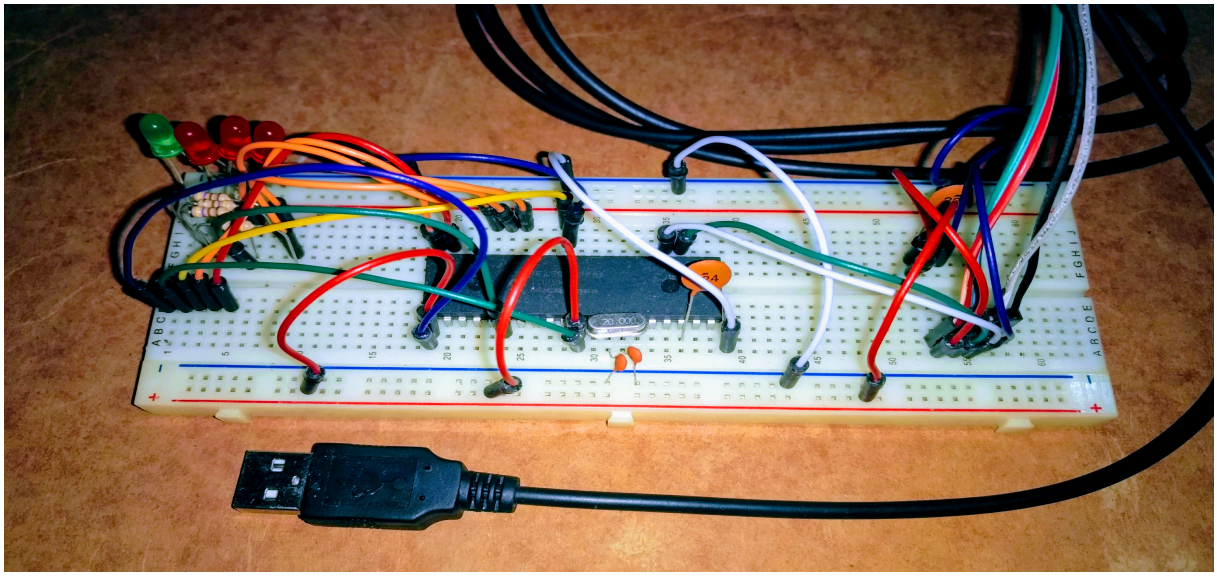
Figura 21 – Diagrama USB - Solução



Fonte: Adaptado de *print screen* do datasheet do microcontrolador.

Percebeu-se que pino 18 (Vusb) é ligado internamente a um regular de tensão para 3.3V da USB e aos pinos D+ e D-, ambos responsáveis pela transmissão de informações de dados, é claro, pela transmissão dos descritores. Dando uma lida a mais no datasheet, foi descoberto que era necessário um capacitor cerâmico externo de 220 nF com um dos terminais no pino 18 e o outro no terra (*ground*). Com isso o problema foi solucionado. A figura 22 ilustra o circuito finalizado.

Figura 22 – Circuito Físico



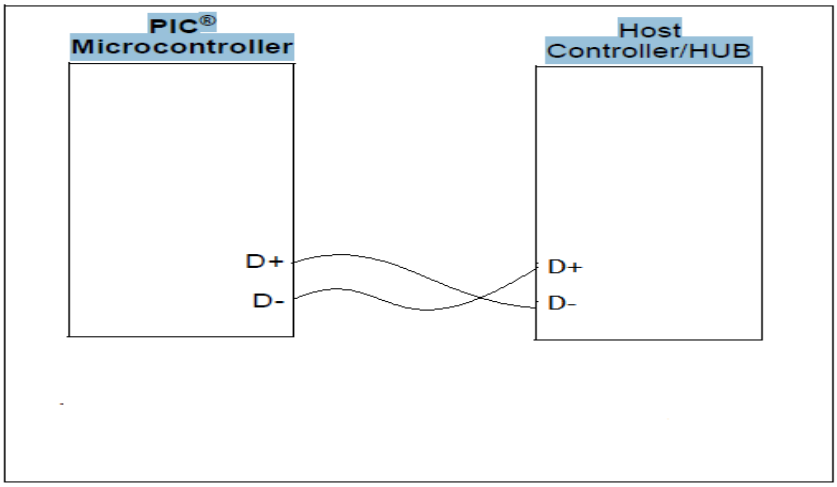
Fonte: Adaptado de *print screen* da tela do Proteus.

Lembrando que o circuito físico modelado nesse trabalho é o básico para comunicação USB HID, ajustado para o exemplo teste descrito no tópico 7 da subseção 4.2.1, comunicável ao sistema web detalhado na seção 4.1, com intenção de propor uma implementação de protocolo de comunicação do mesmo com o chip.

Para a montagem de um circuito USB com o microcontrolador escolhido deve se certificar de apenas 3 passos principais:

- Que o pino Vusb possui o capacitor cerâmico de 220pF.
- Que os capacitores cerâmicos do oscilador estão corretos para a sua frequência.
- Que o cruzamento do D- e D+ do host com o dispositivo, seja realizado com a seguinte configuração, ilustrada na figura 23.

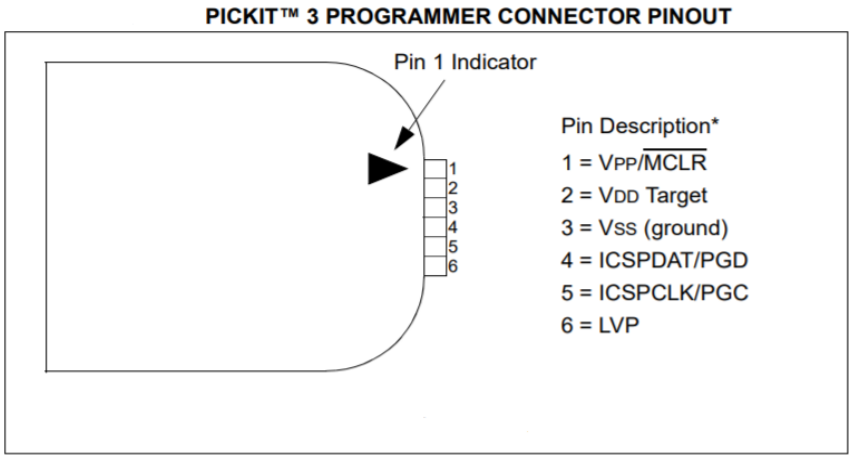
Figura 23 – Esquema cruzamento D+ e D-



Fonte: Adaptado de *print screen* da tela do datasheet do microcontrolador.

4. **Gravação do programa no microcontrolador (embarcar)**: Para a gravação foi utilizado o gravador PICKit3, o qual o MPLAx suporta sem problemas. Na figura 24 os pinos necessários para a gravação do programa. Para efetuar a gravação basta que os pinos do gravador com seus respectivos correspondentes no chip sejam ligados entre si.

Figura 24 – Esquema para a gravação



Fonte: Adaptado de *print screen* do datasheet do PICKit3.

Os pinos descritos estão também no microcontrolador, porém, ambos não apresentam essas configuração, um ao lado do outro, os mesmos estão espalhados em posições distintas no chip. Para facilitar, nesse projeto com o uso de fios (*jumpers*), os mesmos foram organizados lado a lado, na mesma configuração do gravador.

5 Conclusão

“As palavras fogem quando precisamos delas e sobram quando não pretendemos usá-las.”
Carlos Drummond de Andrade

O problema descrito na seção 1.2 foi resolvido como demonstrado no capítulo 4. Nesse capítulo foi descrito o desenvolvimento de um sistema web e um protocolo de comunicação USB, identificando suas limitações e comparado-o com o sistema original do braço robótico Robix Rascal. A comparação deu-se por meio de uma abordagem simples às funcionalidades, tanto da implementação desenvolvida nesse trabalho (ROPIC admin), quanto do sistema do Robix. Desta forma cumpriu-se o objetivo geral e os objetivos específicos "Identificar as limitações da implementação apresentada nesse trabalho" e "Propor uma implementação de protocolo de comunicação: execução de tarefas passo a passo e execução instantânea".

5.1 Contribuições

- Proporcionar desenvolvimentos futuros como a implementação de uma biblioteca embarcável para o controle do braço robótico;
- Permitir e explicitar a integração de sistema web e um protocolo de comunicação;
- Proporcionar e orientar aprimoramentos de forma a permitir que mais funcionalidades de controle de braços robóticos estejam disponíveis na implementação proposta.

5.2 Trabalhos Futuros

- A produção de uma biblioteca embarcável ao microcontrolador do circuito controlador e o uso de um drive PWM/Servo com pelo menos oito canais, com o intuito de controlar de forma paralela e/ou serial, os servos motores do braço, visto que o microcontrolador utilizado nesse trabalho possui apenas dois canais PWM, o que obrigaria a utilização de outros recursos (ex: timers) para o controle.
- Vale ressaltar que, uma vez que o produto, "Sistema Web", foi desenvolvido de forma modularizada, o mesmo pode ser utilizado para facilitar outros trabalhos a executar testes que venham contribuir com o manuseio de tarefas no braço robótico.

Referências

ANGULARJS.ORG. *What Is AngularJS?* 2017. Disponível em: <https://docs.angularjs.org/guide/introduction>. Citado na página 36.

CONSULT, A. *Apostila de Introdução a Robótica*. [S.l.]: Recife, 1995. Citado na página 17.

DJANGOPROJECT.COM. *O Django facilita a criação de melhores aplicativos da Web com mais rapidez e com menos código.* 2017. Disponível em: <https://www.djangoproject.com/>. Citado na página 35.

FESTINGER, L. *Teoria da dissonância cognitiva*. [S.l.]: Zahar, 1975. Citado na página 13.

GONÇALVES, P. C. Protótipo de um robô móvel de baixo custo para uso educacional. *Trabalho de Pós-Graduação*, 2007. Citado na página 33.

HALFPAP, D. M. et al. Um modelo de consciência para aplicação em artefatos inteligentes. Florianópolis, SC, 2005. Citado na página 18.

JONES, J. L.; FLYNN, A. M. *Mobile robots: inspiration to implementation*. [S.l.]: AK Peters, Ltd., 1993. Citado na página 13.

JR, F. L. et al. Sistema de informatização do processo de aquisição dos dados dos agentes de endemias-siade: Módulo aplicação web. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2015. Citado nas páginas 26 e 27.

LAZZARIM, J. C. Construção de um manipulador robótico de baixo custo para ensino. p. 45, apr 2012. Citado na página 33.

MAISONNETTE, R. A utilização dos recursos informatizados a partir de uma relação inventiva com a máquina: a robótica educativa. *PROINFO-Programa Nacional de Informática na Educação, Curitiba-PR*, p. 35, 2002. Citado na página 13.

MICROCHIP, I. T. *PIC18F2455/2550/4455/4550 Data Sheet*. [S.l.], 2009. Citado nas páginas 42 e 44.

MURPHY, R. *Introduction to AI robotics*. [S.l.]: MIT press, 2000. Citado na página 20.

OTTONI, . A. L. C. *Material de estudo - INTRODUÇÃO À ROBÓTICA*. 2010. Disponível em: <http://www.ufsj.edu.br/portal2-repositorio/File/orcv/materialdeestudo_introducaoarobotica.pdf>. Citado na página 17.

PAZOS, F. *Automação de sistema 6 robótica*. [S.l.]: Axel Books, 2002. Citado nas páginas 17 e 19.

PEREIRA, G. Q. O uso da robótica educacional no ensino fundamental: relatos de um experimento. p. 66, 2010. Citado na página 32.

POZZEBON, L. B. F. E. Robótica no processo de ensino e aprendizagem. In: ICBL2013 – INTERNATIONAL CONFERENCE ON INTERACTIVE COMPUTER AIDED BLENDED LEARNING. http://www.icbl-conference.org/proceedings/2013/papers/Contribution42$_{a}.pdf, 2013. Citadonapágina13$.

RUSSELL, S.; NORVIG, P.; SOUZA, V. D. de. *Inteligência artificial: tradução da segunda edição*. [S.l.]: Elsevier, 2004. Citado nas páginas 18 e 19.

SANTOS, L. Sistema de comunicação usb com microcontrolador. *Monografia para Graduação em Engenharia da Computação, UPE. Pernambuco*, 2009. Citado nas páginas 28 e 29.

SANTOS, V. M. Robótica industrial. *Departamento de Engenharia Mecânica*, 2004. Citado na página 13.

SANTOS, V. P. de A. Motor de passo. Julho 2008. Citado na página 23.

SILVA, A. d. *RoboEduc: uma metodologia de aprendizado com robótica educacional. 2009. 127 f.* 2009. Tese (Doutorado) — Tese (Doutorado em Engenharia Computacional)–Pós-graduação em Engenharia Elétrica. Universidade Federal do Rio Grande do Norte, Natal, 2009. Citado na página 13.

SILVA, A. F. da. *RoboEduc: uma metodologia de aprendizado com robótica educacional*. apr 2009. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, apr 2009. Citado nas páginas 17, 18, 19, 22 e 32.

SQLITE.ORG. *Sobre o SQLite*. 2017. Disponível em: <https://www.sqlite.org/index.html>. Citado na página 36.

USB.ORG. *USB 2.0 Documents Index*. 2017. Disponível em: <http://www.usb.org/developers/docs/usb20_docs/>. Citado nas páginas 29, 30 e 58.

VISUALSTUDIO.COM. *IDE do Visual Studio - Ambiente de desenvolvimento integrado e cheio de recursos (IDE) para Android, iOS, Windows, Web e nuvem*. 2017. Disponível em: <https://www.visualstudio.com/pt-br/vs/>. Citado na página 36.

VOLK, M. B. D. S. et al. Unioeste–universidade estadual do oeste do paraná centro de ciências exatas e tecnologicas programa de pós-graduação em engenharia agrícola. 2005. Citado na página 23.

WALAC.GITHUB.IO. *PyUSB - Acesso USB em Python*. 2017. Disponível em: <https://walac.github.io/pyusb/>. Citado na página 36.

ZILLI, S. do R. *A robótica educacional no ensino fundamental: Pespecti- vas e práticas*. oct 2004. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, oct 2004. Citado nas páginas 13 e 31.

Apêndices

APÊNDICE A – Pinos e características do microcontrolador

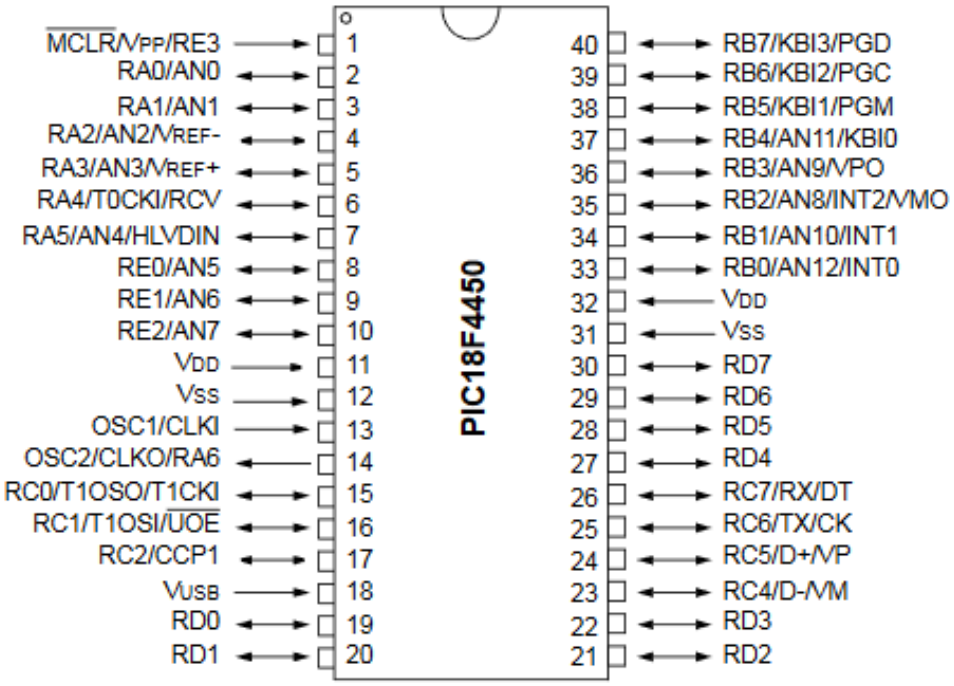
Resumo do hardware do microcontrolador PIC18F4550:

PIC18F4550	
Característica	Descrição
Tipo de Memória do Programa	Flash
Tamanho da Memória do Programa (KB)	32
Velocidade da CPU (MIPS)	12
Tamanho da Memória RAM	2,048 Bytes
Tamanho da Memória EEPROM de dados	256 Bytes
Periféricos de Comunicação Digital	1 - UART 1 - A/E/USART 1 - SPI 1 - I2C 1 - MSSP(SPI/I2C)
Capture/Compare/Periféricos PWM	1 CCP, 1 ECCP
Temporizadores	1 x 8-bit, 3 x 16-bit
Conversores A/D	13 canais, 10-bit
Comparadores	2
USB (canal, tipo)	1, FS Device, USB 2.0
Faixa de Temperatura Operacional (C)	-40 até 85
Faixa de Tensão Operacional (V)	2 até 5.5
Quantidade de Pinos	40

Tabela 1 – Descrição Microcontrolador PIC18F4550.

A figura 25 apresenta todos os pinos do microcontrolador. As setas representam o tipo do pino, entrada e/ou saída.

Figura 25 – Pinagem PIC18F4550 [1]



Fonte: *print screen* do *datasheet* do Microcontrolador.

[1] Figura 25:
Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/39760d.pdf.
Acesso em Out. 2016

APÊNDICE B – Configurações do microcontrolador

Essas configurações podem ser geradas por meio de um assistente de configuração existente na própria IDE MPLABx.

Configuração - PIC18F4550	
Registrador de Configuração	**Valor**
#pragma config PLLDIV	5
#pragma config CPUDIV	OSC1_PLL2
#pragma config USBDIV	2
#pragma config FOSC	HSPLL_HS
#pragma config FCMEN	OFF
#pragma config PWRT	OFF
#pragma config BOR	OFF
#pragma config BORV	3
#pragma config VREGEN	ON
#pragma config WDT	OFF
#pragma config WDTPS	32768
#pragma config CCP2MX	ON
#pragma config PBADEN	OFF
#pragma config LPT1OSC	OFF
#pragma config MCLRE	OFF
#pragma config STVREN	ON
#pragma config LVP	OFF
#pragma config ICPRT	OFF
#pragma config XINST	OFF
#pragma config DEBUG	OFF
#pragma config CP0,CP1,CP2,CP3	OFF
#pragma config CPB	OFF
#pragma config CPD	OFF
#pragma config WRT0,WRT1,WRT2,WRT3	OFF
#pragma config WRTC	OFF
#pragma config WRTB	OFF
#pragma config WRTD	OFF
#pragma config EBTR0,EBTR1,EBTR2,EBTR3	OFF
#pragma config EBTRB	OFF

Tabela 2 – Cabeçalho de Configuração - PIC18F4550.

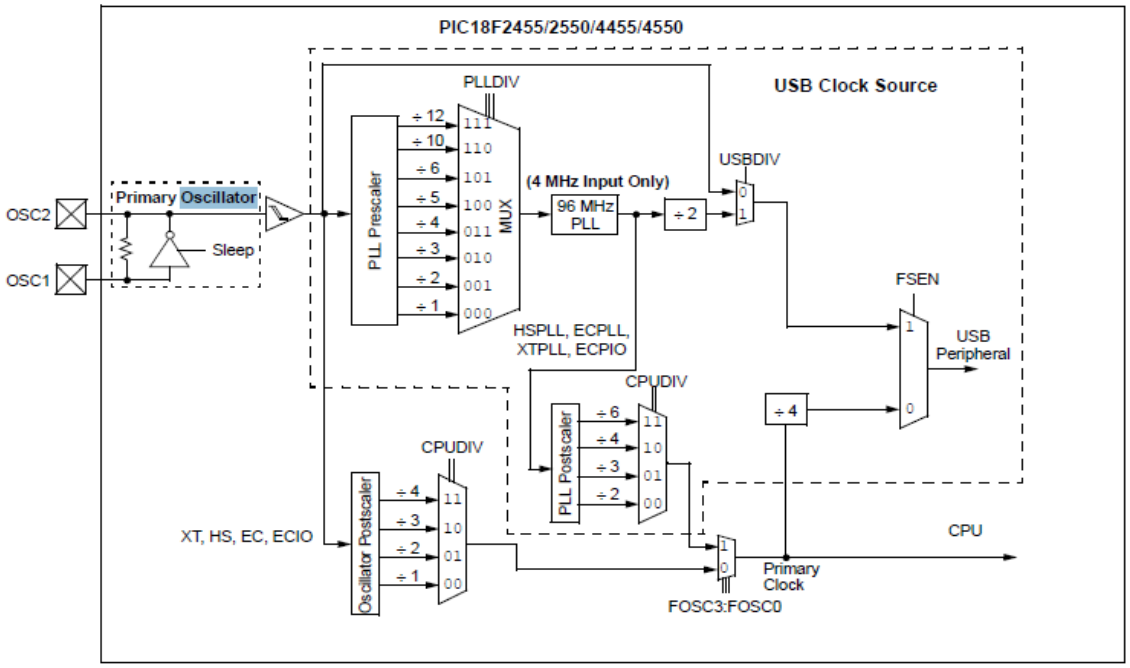
Todos as configurações recebem a assinatura "#pragma config VALOR", conforme a tabela 2. O token "pragma" indica para o compilador que as informações "REGISTRADOR DE CONFIGURAÇÃO = VALOR" serão armazenadas em um endereço específico na memória, endereços de configuração, devido ao token consequente "config".

O chip conta com um circuito PLL (Phase Locked Loop - Laço Travado em Fase) que

multiplica o valor da frequência desse oscilador, o que facilita obter os 48Mhz para a USB funcionar em modo *Full-Speed*. Na figura 26 pode se observar os 4 registradores seguintes: PLLDIV, FOSC, CPUDIV, e USBDIV.

Para configurar as frequências da CPU e da USB, do chip, precisou-se entender cada um desses registradores. Nesse trabalho foi utilizado um oscilador de cristal (componente eletrônico) de 20Mhz, porém o projeto requer um clock de 48Mhz para a USB, para que a mesma funcione em alta velocidade, para se obter tal frequência, tendo como base o *datasheet* do chip, chegasse a conclusão que: o PLLDIV deve ser 5 (100'b) devido a frequência da oscilador ser 20Mhz, o FOSC deve ser 14 ou 15 (111x'b) devido o oscilador(Crystal) ser do tipo HSPLL, o CPUDIV deve ser 0 (00'b) para a CPU tenha a mesma frequência da USB e por fim, a USBDIV deve ser 1 para que obter os 48Mhz desejados. Tais configurações estarão presente no cabeçalho de configuração, descrito anteriormente.

Figura 26 – Diagrama do clock para o oscilador primário [1]



Fonte: Adaptado de *print screen* do *datasheet* do microcontrolador.

[1] Figura 26:
Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/39760d.pdf.
Acesso em Out. 2016

APÊNDICE C – Descritores USB

Informação para configuração de descritores usados na USB.

Cada dispositivo USB possui apenas um descritor de dispositivo, tabela 3, que contém informações gerais do mesmo. Tais parâmetros de configuração são os primeiros a serem coletados pelo computador quando o dispositivo USB é conectado (USB.ORG, 2017).

Caso queira implementar a versão 1.1 da USB, o parâmetro bcdUSB deve ter o valor alterado para 0110h. Neste caso está configurado para a versão 2.0.

DESCRITOR DE DISPOSITIVO			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	12h	Tamanho do descritor
bDescriptorType	1	01h	Tipo de descritor
bcdUSB	2	0200h	Versão USB utilizada
bDeviceClass	1	00h	Classe do dispositivo
bDeviceSubClass	1	00h	Sub Classe do dispositivo
bDeviceProtocol	1	00h	Protocolo utilizado
bMaxPacketSize0	1	08h	Tamanho max. do pacote
idVendor	2	04D8h	Id do fabricante
idProduct	2	003Fh	Id do produto
bcdDevice	2	0002h	Versão do produto
iManufacturer	1	01h	Informações do produto
iProduct	1	02h	Informações do produto
iSerialNumber	1	00h	Nº de Serie do produto
bNumConfigurations	1	01h	Nº de configurações

Tabela 3 – Descritor de dispositivo.

No descritor de configuração, tabela 4, devem ser observados dois parâmetros principais: o bmAttributes e o bMaxPower. Como a tabela acima mostra, o bmAttributes possui o tamanho de 1Byte = 8bits. Os bits 0, 1, 2, 3, 4 e 7 são reservados, o bit 5 é responsável pelo suporte ao *Remote Wakeup*, uma função que desativa o dispositivo USB enquanto o mesmo não está sendo utilizado, a mesma não está sendo usada no projeto. O bit 6 é o bit *Self Powered*, o que torna o bmAttributes um parâmetro importante, tal bit é responsável em indicar o tipo de alimentação do circuito, deve-se atentar que esse esteja em nível 1 para que alimentação seja por meio da USB.

Por fim, o bMaxPower é um dos principais nesse descritor, justamente porque é o byte responsável em indicar a corrente máxima requerida na comunicação USB, o padrão é 50 quando o *Self Powered* está em nível 1.

DESCRITOR DE CONFIGURAÇÃO			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	09h	Tamanho do descritor de configuração
bDescriptorType	1	02h	Tipo de configuração
wTotalLength	1	29h	Tamanho de todas as configurações
bNumInterfaces	1	01h	Nº de interfaces
bConfigurationValue	1	01h	Id da configuração
iConfiguration	1	00h	Índice do descritor de string da configuração
bmAttributes	1	C0h	Definição de vários atributos
bMaxPower	1	50	Máxima corrente requerida (maximum milliamperes/2)

Tabela 4 – Descritor de configuração.

No descritor de interface é atribuída basicamente a classe da interface de transmissão de dados em bInterfaceClass e o número de *endpoints* em bNumEndpoints. Durante a configuração de dispositivos USB é possível atribuir várias interfaces de comunicação, inclusive de classes diferentes. Nesse trabalho foi implementado duas interfaces: uma geral para o controle do dispositivo, tabela 5, e uma específica da classe HID para transmissão de dados, tabela 6.

DESCRITOR DE INTERFACE DE CONTROLE			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	09h	Tamanho do descritor
bDescriptorType	1	04h	Tipo de descritor
bInterfaceNumber	1	00h	Id da interface
bAlternateSetting	1	00h	Configuração alternativa
bNumEndpoints	1	02h	Numero de endpoits
bInterfaceClass	1	03h	Classe da interface
bInterfaceSubClass	1	00h	Sub classe
bInterfaceProtocol	1	00h	Protocolo utilizado
iInterface	1	00h	Índice do descritor de string da interface

Tabela 5 – Descritor de interface de controle.

DESCRITOR DE INTERFACE HID			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	09h	Tamanho do descritor HID
bDescriptorType	1	21h	Tipo de configuração
bcdHID	2	1101h	Número de versão HID
bCountryCode	1	00h	Código do país do hardware.
bNumDescriptors	1	01h	Número de descritores
bDescriptorType	1	22h	Tipo de descritor
bDescriptorLength	2	29h	Tamanho em bytes do descritor

Tabela 6 – Descritor da interface de transmissão de dados do tipo HID.

Os descritores de endpoints definem o tipo de transferência de dados suportada pelo endpoint (controle, massa, interrupção e isossíncrona), a direção dos dados, o tamanho máximo do pacote de dados, e o intervalo de consulta aos endpoints de interrupção e isócronos. Nesse projeto foram definidos dois endpoints. O endpoint 0, usado para enviar os dados e o endpoint 1 que será usado para receber os dados, tabelas 7 e 8, respectivamente.

DESCRITOR DE ENDPOINT 0			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	07h	Tamanho do descritor
bDescriptorType	1	05h	Tipo de descritor
bEndpointAddress	1	81h	Endereço
bmAttributes	1	03h	Tipo de transferência
wMaxPacketSize	2	40h	Tamanho do pacote
bInterval	1	01h	Intervalo temporal (ms)

Tabela 7 – Descritor de endpoint 0.

DESCRITOR DE ENDPOINT 1			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	07h	Tamanho do descritor
bDescriptorType	1	05h	Tipo de descritor
bEndpointAddress	1	01h	Endereço
bmAttributes	1	03h	Tipo de transferência
wMaxPacketSize	2	40h	Tamanho do pacote
bInterval	1	01h	Intervalo temporal (ms)

Tabela 8 – Descritor de endpoint 1.

Abaixo segue os descritores de string, tabelas 9 e 10. Nesses descritores são atribuídos informações legíveis por humanos, mesmo sendo opcionais, são de extrema importância. Nesse projeto foram definido três descritores de string. Um para indicar o idioma suportado pelo dispositivo, outro para indicar a fabricante do dispositivo e por ultimo um para indicar qual o produto.

DESCRITOR DE STRING (PRODUTO)			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	3Ch	Tamanho do descritor
bDscType	1	03h	Tipo de descritor
bString	N	TCC-NIL MARTINS (ROPIC) V1.0	Nome do produto

Tabela 9 – Descritor de string (produto).

DESCRITOR DE STRING (FABRICANTE)			
ITEM	**TAM**	**VALOR**	**DESCRIÇÃO**
bLength	1	34h	Tamanho do descritor
bDscType	1	03h	Tipo de descritor
bString	N	Microchip Technology Inc.	Nome da fabricante

Tabela 10 – Descritor de string (fabricante).

APÊNDICE D – Rotinas USB

Rotinas da biblioteca:

- **void USBInit()**:Inicializa a USB.
- **void USBWorks()**: Executa as tarefas USB conforme o estado do dispositivo.
- **void USBSuspend()**: Coloca a USB no estado de suspensão.
- **void USBWakeUP_Suspend()**: Retira a USB do estado de suspensão.
- **void USBChecker_Stall()**: Manipula erros Stall, verificando se Endpoint 0 emitiu algum pacote Stall.
- **void USBChecker_Request_Std()**: Verifica se a USB possui uma solicitação de transferência padrão (via interface padrão).
- **void USBChecker_Request_Hid()**: Verifica se a USB possui uma solicitação de transferência HID (via interface HID).
- **void USBTransfer_ctrlTypeChecker()**: Faz o controle de transferências, verifica se a transação é de entrada ou saída ou de configuração, encaminhando para a função responsável.
- **void USBTransfer_ctrlSetup()**: Encaminha para o controle de transferência de configuração do dispositivo.
- **void USBFinisher_ctrlSetup()**: Finaliza os processos de controle de transferência de configuração do dispositivo.
- **void USBPreparer_ctrlSetup()**: Prepara para a próxima transferência de configuração.
- **void USBForwards_Transfer_ctrlReceive()**: Encaminha para o controle de transferências de recepção de dados, sentindo host -> dispositivo.
- **void USBForwards_Transfer_ctrlTransmit()**: Encaminha para o controle de transferências de envio de dados, sentindo dispositivo -> host.
- **void USBForwards_dataReceive()**: Encaminha para a recepção de dados.
- **void USBForwards_dataSend()**: Encaminha para o envio de dados.
- **void USBTransfer_ctrlReceive()**: Responsável por fazer o controle de transferências de recepção de dados.
- **void USBTransfer_ctrlTransmit()**: Responsável por fazer o controle de transferências de envio de dados.

- **void USB_dataReceive()**: Recebe os dados.
- **void USB_dataSend()**: Envia os dados.

Rotinas comuns:

- **void main()**: Função principal, onde se inicial todos os processos de comunicação USB.
- **void TestInit()**: Criada para configurar e inicializar o teste da comunicação USB.
- **void TestWorks()**: Criada para executar o teste da comunicação USB, tal será detalhado abaixo.